Secured Intrusion Detection System Infrastructure

S. Mrdović*, E. Zajko**
* University of Sarajevo/Faculty of Electrical Engineering, Sarajevo, Bosnia and Herzegovina
sasa.mrdovic@etf.unsa.ba
** University of Sarajevo/Faculty of Electrical Engineering, Sarajevo, Bosnia and Herzegovina
ernedin.zajko@etf.unsa.ba

***Abstract*—This paper will present building of secured intrusion detection system (IDS) infrastructure. For its function IDS is often the first target of intruders and must be properly secured. Main components of IDS and principles for their hardening will be explained. Application of these principles in practice will be shown on the secured IDS infrastructure that will be built using open source products.**

***Keywords* – intrusion detection, network security**

I. INTRODUCTION

Intrusion detection systems (IDS) are becoming standard part of comprehensive security system. They are feature of the defense-in-depth strategy. A firewall is an essential and important part of network security but it does not have the ability to detect hostile intent. Unlike a firewall, an intrusion detection system has the ability to evaluate solitary packets and generate an alarm if it detects a packet with hostile potential.

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. Intrusion detection systems fall into two basic categories: signature-based intrusion detection systems and anomaly detection systems. Intruders have signatures, like computer viruses, that can be detected using software. IDS tries to find data packets that contain any known intrusion-related signatures or anomalies related to Internet protocols. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts. Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts. In some cases these methods produce better results compared to signature-based IDS. Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it.

Network IDS (NIDS) are intrusion detection systems that capture data packets traveling on the network media (cables, wireless) and match them to a database of signatures. Depending upon whether a packet is matched with an intruder signature, an alert is generated or the packet is logged to a file or database.

Host-based intrusion detection systems or HIDS are installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity. Some of these systems are reactive, meaning that they inform you only when something has happened. Some HIDS are proactive; they can sniff the network traffic coming to a particular host on which the HIDS is installed and alert you in real time.

There is a great deal of work that is currently being performed in the area of intrusion detection. Much of the work centers around improvement in the ability of systems to detect attacks and the speed of network traffic that can be handled.

This paper will concentrate on IDS security, area less explored in recent papers. IDS acts as a guard monitoring for suspicious activity. If guard is removed or prevented from seeing intrusion it is useless. We will present principles and build Network IDS based on those principles using open source tools. The paper does not consider rules used do detect attacks, just secured infrastructure as basis for building efficient IDS.

II. IDS COMPONENTS

There are number of different ID system designs. The Common Intrusion Detection Framework (CIDF) [1] defines a set of components that together define an intrusion detection system. These components include event generators (``E-boxes''), analysis engines (``A-boxes''), storage mechanisms (``D-boxes''), and even countermeasures (``C-boxes''). A CIDF component can be a software package in and of itself, or part of a larger system. Figure 1 shows the manner in which each of these components relates.

As Ptacek and Newsham [2] pointed out, each component identified by the CIDF model has unique security implications, and can be attacked for different reasons.

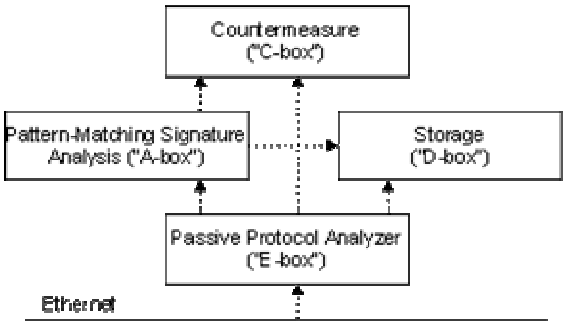

Figure 1. Common Intrusion Detection Framework (CIDF) Components

As the only inputs of raw data into the system, E-boxes act as the eyes and ears of an IDS. An attack against the event generation capabilities of an IDS blinds it to what's actually happening in the system it's monitoring. For example, an attack against the E-box of a network IDS could prevent it from obtaining packets off the network, or from appropriately decoding these packets.

Some intrusion detection systems rely on sophisticated analysis to provide security information. In such systems, the reliability of the A-box components used is important because an attacker that knows how to fool them can evade detection --- and complicated analytical techniques may provide many avenues of attack. On the other hand, overly simplistic systems may fail to detect attackers that intentionally mask their attacks with complex, coordinated system interactions from multiple hosts [3].

The need for reliable data storage is obvious. An attacker that can subvert the D-box components of an IDS can prevent it from recording the details of her attack; poorly implemented data storage techniques can even allow sophisticated attackers to alter recorded information after an attack has been detected, eliminating its forensic value.

The C-box capability can also be attacked. If a network relies on these countermeasures for protection, an attacker who knows how to thwart the C-box can continue attacking the network, immune to the safety measures employed by the system. More importantly, countermeasure capabilities can be fooled into reacting against legitimate usage of the network --- in this case, the IDS can actually be turned against the network using it (often undetectably).

It is apparent that there are many different points at which an intrusion detection system can be attacked. This paper will try to compile advices on securing each of the components of IDS and apply them on distributed network intrusion detection system that we build.

There are other problems with the use of passive protocol analysis as an event-generation source for signature-analysis intrusion detection systems. This paper does not consider those problems since they are inherent to NIDS design and can not be removed by improved IDS security.

III. SECURED NIDS

A. Architecture

We build distributed IDS. Term distributed IDS is used to indicate system in which more then one sensor is used to collect network traffic. For each network segment that we want to detect intrusion we place a sensor. Data from all sensors is sent to a central location and stored and analyzed from there. Distributed IDS can be centrally managed and cover all important parts of the network. Typical places where sensors are positioned include Internet entry points, just inside routers and/or firewalls, and DMZ. Those positions enable detection of external intrusions. In order to detect internal intrusions sensors must be placed on internal network segments as well.

From the aspect of securing IDS selection of network segments to be monitored does not have too much influence on the way the sensor is protected. If network segment warrants monitoring sensor must be considered to be working in unfriendly environment and all of them must be secured in the same way. Position of central IDS storage and connection of storage with sensors and management consoles has major impact on the security of the IDS. We selected to implement the system where all sensors have two network cards. One connected to the network segment being monitored and the other one connected to isolated network segment dedicated to IDS. This isolated network segment includes central storage server, internal sides of all sensors and management console. By isolating IDS storage and management system we reduce its exposure to external attacks. We also use secured connection within this network segment in spirit of layered security. In the event that intruders are somehow able to monitor traffic on this segment secured connections would prevent them from understanding and modifying data being exchanged among components of IDS. Described architecture is shown on Figure 2.

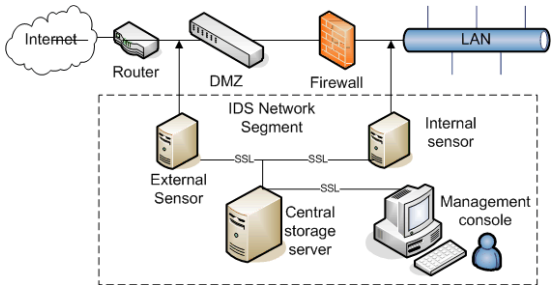

Figure 2. NIDS architecture

B. Sensors

In our implementation sensors are computers that implement two CIDF defined components: event generators (E – boxes) and analysis engines (A – boxes). We will describe steps needed and taken in order to harden sensors and secure their functions within IDS.

Since event generators monitor traffic that might be malicious they are directly exposed to attacks. There are several steps that can be taken to harden sensor boxes. First of all is installation of fully patched operating system. The network card that is connected to network segment being monitored must be put in promiscuous mode so it can listen to all the traffic on that segment. That card should not be configured with an IP address, so it will be invisible to hosts on that network. This is commonly referred to as a stealth interface. Keeping the listening interface invisible to the other systems on the network makes keeping the sensor secure much easier [4]. Sensor computer will be used only as NIDS sensor, so all unnecessary services should be disabled.

We built our sensor boxes on AMD Athlon XP 2500+ based machines, with 1 GB RAM and 80 GB hard disk, that we had available, but sensor could be implemented on much older hardware platform [5] [6]. Operating system installed for sensors was OpenBSD 3.7 with minimal distribution set, no X, and all the patches applied. OpenBSD is operating system oriented to security and considered to be safest choice for implementations that need to be very secure. OS was secured as described in [7]. We disabled all services except for sshd [8], that we configured with public key authentication. We installed Snort 2.1.2. [9], open source network IDS, to be used as analysis engine. We configured Snort to run chrooted and drop privileges to unprivileged user with completely

locked out account. We installed MySQL 4.0.23 [10] client to enable Snort to log events to MySQL database located on central storage server. MySQL is open source database. For protection of MySQL communication between MySQL client on sensor and MySQL server on central storage we installed Stunnel 4.08. [11]. Stunnel is open source program that allows encryption of arbitrary TCP connections inside SSL. Sensor computer is firewalled, using Open BSD pf, to prevent any inbound connections except for ssh from management console.

In this manner we built secured sensor invisible from the network segment it monitors that communicates securely with central storage server that will be described next.

C. Central storage system

Central storage server is located on isolated network segment dedicated to IDS. This makes it less exposed to direct attack. Nevertheless, security should be implemented at host level as well. Fully patched and hardened OS creates base for secure server. Next step is determining what services should this server provide and corresponding applications that provide those services. We need this server to be database server that stores events generated by sensors so we must install and secure database server application and provide enough space to store data. We also need means to review events from database from management console. Web based application is usual and convenient way of doing this so we must install and secure web server and web application.

Hardware platform on which we built central storage server is identical to the one used for sensors. Important thing to keep in mind when selecting hardware is disk space needed to store alerts database. Space requirement depends on number of sensors and amount of traffic. Installed operating system was, same as for sensors, OpenBSD 3.7, and was hardened in the same way. For database server we used MySQL 4.0.23 server and secured it as described in [12]. For web server we used Apache 1.3 that comes as part of basic OpenBSD installation. OpenBSD Apache is already chrooted for added security. Additional security measures were implemented as described in [7]. SSL is enabled and server certificate issued locally using OpenSSL [13], open source toolkit for implementing SSL. This enables SSL secured connection from management console to this web server. Application we used for presenting IDS events in database is ACID (Analysis Console for Intrusion Detection) 0.9.6 [14], open source application specially designed for this purpose. ACID has no authentication built in, so we used Apache authentication with mod_ssl based on X.509 certificates to allow connections only to user with certificate connecting from management console [15]. As with sensor we installed Stunnel to SSL protect MySQL communication with sensors. Server management is enabled through sshd configured with public key authentication. OpenBSD firewall pf is used to allow incoming connections only from sensors for Stunnel protected MySQL traffic and ssh from management console.

Central storage server built and configured in this way does all of its functions within intrusion detection system but in a very secure manner.

D. Management console

Management console is the computer used to monitor and control IDS. It is the only computer allowed to connect to sensors and central storage server. It does not need any special software except for the web browser.

We used the same hardware and operating system as for the sensors and the central storage server. Since user of this console needs to be able to ssh into sensors and central storage server we created private-public key pair and distributed public key to sensors and server, and protected private key on console with pass phrase as described in [16]. Login to ACID is allowed only through SSL with client certificate we created. We also imported server certificate into web browser on console to prevent browser warnings.

E. Integration

Figure 3. shows integrated system with all important components. All communications are encrypted and authenticated with public key certificates. On all boxes only needed services are installed and enabled, and all of them are secured. It is of utmost importance to insure physical security of all IDS computers, since without physical security all other protections are useless.

IDS built in this way can be made portable, since the only thing that is different for different networks being monitored are sensor rules. This enables preparing complete IDS setup in the lab and makes installation on site just a matter of plugging sensor monitoring interface at right points in the network. Such configured and ready to go IDS could be a product on hot IDS market.

IV. CONCLUSION

We described IDS components based on CIDF. We explained security implications for each of the components. We proposed secured distributed network IDS architecture with physical components corresponding to CIDF. Principles to harden each component were

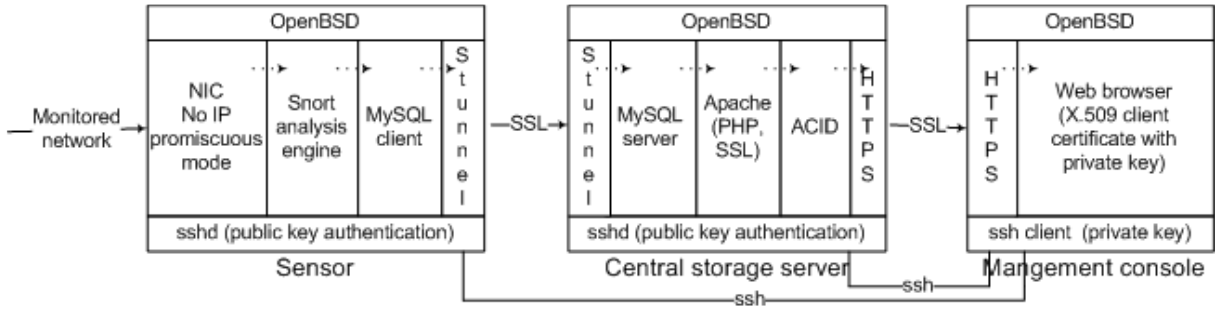

Figure 3. NIDS components and interactions

explained and applied using standard hardware and open source software. We built secured intrusion detection system infrastructure. It provides safe platform to implement intrusion detection logic based on IDS policy. We did not consider any rules to detect network attacks.

Further steps might include work on defining intrusion detection rules for a particular environment. In the area of management it might be worth exploring ways of having management console located on different network segment and providing secure communication with central storage server and sensors. Host intrusion detection systems could be installed on all DMZ servers as well as all LAN servers. Distributed Host IDS programs, like Osiris [17], could be made part of integrated IDS. Inclusion of *honeypots* [18] using tools like Honeyd [19] would be a nice addition to system. Counter measures, CIDF C-boxes, might be activated turning system into intrusion prevention system (IPS). This could be achieved by connecting firewall, OpenBSD pf, to system and changing filtering rules based on detected intrusions.

Area of intrusion detection is in its full development now and there are many new ideas being considered. We tried to provide secure foundation for implementation of those ideas.

REFERENCES

[1] Staniford-Chen, S., B. Tung, and D. Schnackenberg. "The common intrusion detection frame-work (CIDF)." In Proceedings of the Information Survivability Workshop, 1998.

[2] T. Ptacek and T. Newsham. Insertion, evasion, and denial of service: eluding network intrusion detection. Technical report. Secure Networks Inc., January 1998.

[3] N. F. Puketza, K. Zhang, M. Chung, B. Mukherjee and R. A. Olsson, ``A Methodology for Testing Intrusion Detection Systems," IEEE Transactions on Software Engineering, vol. 22, pp. 719-729, October 1996.

[4] K. Cox, C. Gerg, "Managing Security with Snort & IDS Tools", O'Reilly, 2004.

[5] TJ Vanderpoel, "Deploying Open Sourced Network Intrusion Detection for the Enterprise", SANS, 2001, http://www.sans.org/resources/idfaq/open_source.php

[6] M. P. Brennan, "Using Snort For a Distributed Intrusion Detection System", SANS, 2002, http://www.sans.org/rr/whitepapers/detection/352.php

[7] Y. Korff, P. Hope, B. Potter, "Mastering FreeBSD and OpenBSD Security", O'Reilly, 2005.

[8] http://www.openssh.org

[9] http://www.snort.org

[10] http://www.mysql.org

[11] http://www.stunnel.org

[12] http://dev.mysql.com/doc/mysql/en/security.html

[13] http://www.openssl.org

[14] http://www.cert.org/kb/acid

[15] "Client certificates with apache", http://www.garex.net/apache/

[16] "OpenSSH Public Key Authentication" http://cfm.gs.washington.edu/security/ssh/client-pkauth

[17] http://www.hostintegrity.com/osiris/

[18] http://www.honeypots.net/

[19] http://www.honeyd.org/