Kerckhoffs' Principle for Intrusion Detection

Sasa Mrdovic, Branislava Perunicic

***Abstract*—One of the basic principles of cryptography is that the security of a system must depend not on keeping secret the algorithm, but only the key. This principle is known as Kerckhoffs' Principle. In this paper we propose application of this principle in intrusion detection systems. The fact that attackers know the intrusion detection algorithm will not help them if there is a secret key for each implementation that makes it different enough from the others. Implementation of network packet payload anomaly detection IDS that enables application of the idea is presented. Results for various keys confirm excellent detection capabilities. Proof of concept mimicry attack protection example is provided.**

***Index Terms*—anomaly detection, Kerckhoffs' principle, network intrusion detection, word models.**

I. Introduction

THE history of intrusion detection systems has been compared to an arms race [1]. Similar to any protection system, when the defenders find a method to discover a certain type of attack, the attackers study the method and find ways to get around it. There are numerous examples in the IDS arena and, to name only a few, in academic papers, like [2], which pointed out serious flaws in all NIDS of the time, or more recently [3], which showed how [4], [5], [6], [7] anomaly detection methods could be evaded.

Cryptography put an effective stop to this type of arms race following the principle that the security of a system must depend not on keeping secret the algorithm, but only the key. Modern cryptographic algorithms passed the test of time. Data Encryption Standard (DES) [8] was the US encryption standard and de facto standard in the rest of the world for 25 years, from 1977 to 2002. The DES algorithm was practically never broken. It was replaced with Advanced Encryption Standard (AES) [9] mainly because the DES key was not big enough for current brute force computing power.

A mimicry attack is first defined in [10] as malicious exploit code that mimics the operation of the application thus evading detection by anomaly detectors. This definition was given in the context of host IDS, but the idea applies to network IDS as well. For NIDS, a mimicry attack tries to make attack sessions look like normal ones. This should prevent their identifications as anomalous and therefore detection. Knowing how a detection method creates the model of a normal session could enable attackers to create attack sessions that mimic normal ones. But if there is a secret key for each implementation of the detection method that makes it different enough from the others, attackers will not be able to mimic normal behavior without knowledge of the key.

In this paper, we propose using this cryptography principle for intrusion detection. We present this novel and original idea and one practical implementation. First we implement a base system, simple and efficient state of the art anomaly detection network intrusion detection system. Then we show how a key can be added to the base system detection algorithm, implementing Kerckhoffs' principle. We provide detection results for the base system and keyed systems with proof of concept showing how the proposed implementation prevents mimicry attack.

II. Related work

Payload analysis of network packets to detect intrusions became a necessity. It enables detection of application level attacks. Those attacks make up the majority of current computer attacks [11] for two basic reasons. Most of the current vulnerabilities that create opportunities for attacks are in user applications [12]. Most intrusions at a lower level of the network protocol stack could now be efficiently prevented using properly configured standard network protection equipment like firewalls. Techniques of anomaly-based network intrusion detection have been proposed that analyze payloads of packets and connections [4], [6], [13], [14], [15]. HTTP oriented analysis is given in [16]. Detection of byte sequences that resemble code is proposed in [17] and [18]. Syntax and semantic information is used in [1] and [19]. Single byte frequencies in payload as a basis for the model are used in [7] with improvements in [20]. Recent research uses frequencies of more than one byte, n-grams, [21] or language model based on words, consecutive bytes in payload separated by delimiters [22].

General issues and difficulties with NIDS and evasion ideas are best given in [2]. The first actual mimicry attacks are given in [23] and [24]. Specific ideas on how to avoid anomaly based NIDS detection are given in [25] and [26]. Polymorphic blending attacks that evade most of currently proposed payload analysis method are described in [3].

As we already said, it is an arms race. There are very few ideas for fighting mimicry attacks on anomaly detection NIDS. One practical implementation of protection against general

Manuscript received March 7, 2008.

S. Mrdovic is with the University of Sarajevo, Sarajevo, Bosnia and Herzegovina (e-mail: sasa.mrdovic@ etf.unsa.ba).

B. Perunicic is with the University of Sarajevo, Sarajevo, Bosnia and Herzegovina (e-mail: bdrazenovic@ etf.unsa.ba).

mimicry attacks is presented in [21]. We are not aware of any general approaches like the one we suggest.

III. KERCKHOFFS' PRINCIPLE

In 1883 Kerckhoffs suggested six principles of design of practical cryptographic ciphers [27]. The second of those principles is known as Kerckhoffs' principle and states that the design of a system should not require secrecy, and compromise of the system should not inconvenience the correspondents. Shannon later restated that it should always be assumed that "the enemy knows the system being used" and this statement is known as Shannon's maxim [28]. The idea that security of the cryptographic system rests in the keys, not in the algorithm, is one of the main concepts in modern cryptography. In cryptography, a key can be compromised, and can be replaced with different one without a need to redesign algorithm. This concept, that if there was a compromise of secrecy whatever is compromised should be easily replaceable, applies to security in general. Saltzer and Schroeder [29] include it in their eight principles for the design and implementation of security mechanisms and call it "Open design". They say that design should not be secret and that mechanisms should not depend on the ignorance of potential attackers, but rather on the possession of specific, more easily protected, keys or passwords. In physical security if a door key is lost or stolen, only locks need to be changed, or sometimes a lock can simply be reprogrammed, as in hotels. This is not just more convenient but is also more secure if the key space is large enough.

We suggest applying the same principle when designing intrusion detection systems. Most intrusion detection schemes can be defeated with careful crafting of attack vectors for the particular defense mechanism. If the detection algorithm is known but the actual detection of attack depends on some piece of information unknown to attacker, this makes it very difficult if not impossible to create an attack that will not be detected.

IV. BASE SYSTEM

Our base system detects anomalies in network packet payload. Packet payload analysis divides payload into groups of consecutive bytes, n-grams. It has been shown that using higher order n-grams for payload modeling yields better detection results [21] than using 1-gram model [7]. Selection of optimal length of n-grams is no easy task and the result might depend on the network protocol being modeled and the particular traffic dataset. Using words, byte sequences separated with boundary symbols, for text-based protocols like HTTP, SMTP and FTP achieves marginally less accurate results than modeling with several n-gram lengths at once, but with much smaller computational load [22]. We used words as the basis for our model, following advice given in [22] to limit byte sequence length to 16 bytes in absence of delimiters.

A. Learning

Initial research focused on HTTP protocol with a set, extended from [14], of 20 delimiters we found to be the most appropriate, since they provide the biggest number of meaningful words:

CR LF TAB SPACE , . : / \ & ? = ( ) [ ] " ; < >

We trained our system using supervised learning. Supervised learning should be performed on normal, attack free, traffic. Obtaining a clean data set might be difficult. Collection of normal production network traffic for a period of time does not guarantee that it is attack free and also does not guarantee that all possible normal packets will be seen. The duration of time required to collect enough traffic might also be longer than desired. A detection method should be able to cope with those realistic limitations on learning. A good overview of the machine learning issues and possible solutions is given in [30].

We used a semi-automated process to obtain attack-free traffic from collected network traffic. HTTP traffic collected from our department Web servers was scanned using the signature-based network intrusion detection, Snort [31]. A scan was performed with all rules activated and with the latest signature content of Snort rules obtained from [32]. In this way we were able to obtain traffic that can be considered clean enough for training.

After some experimenting with various amounts of traffic used for training, we found that after learning for 96 hours, the number of learned words increases very little with additional traffic. At this point we concluded that there would be little gain from further training. Fig. 1 shows the dependency of the number of words learned on the number of hours of network traffic used for training.

We parsed only incoming HTTP traffic since we want to detect attack by recognizing incoming traffic that our servers have never seen. This provides for smaller number of words and faster processing without significant negative impact on detection, as will be shown later. Our total number of learned normal words was little over 33 000.

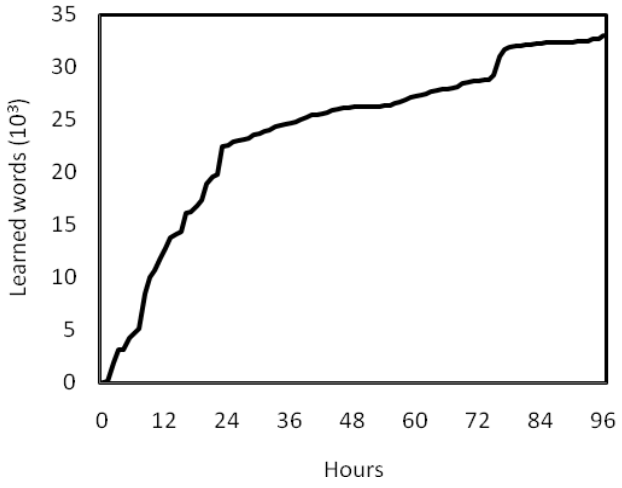

Fig. 1. Number of learned words as a function of hours of network traffic

All words found in normal traffic are stored in a hash table, together with the frequencies with which they appear. At this point word frequencies are not used, but we have certain ideas for future use of those frequencies.

B. Detection

In the detection phase, we tokenize to words incoming HTTP network traffic payloads using the same delimiters used during the learning phase. For each of the words we check if it is in a set of learned words. Then we a use simple formula similar to the one suggested in [21] for packet scoring.

Score = New / Total

New is the number of new words, not in a set of learned words, in a packet. Total is the total number of words in a packet. This score is a measure of packet anomaly. The score is expressed as a percentage with values from 0 to 100. We first tested detection on seven days of traffic we collected during the normal working course of our Web sites. Scores for most of the normal packets were less than ten. There were from one to ten non-attack packets per day that had scores over 20. There was only one exceptional score for a non-malicious packet in seven days, with score of 64.

In the second experiment, we considered how our algorithm scores probes. We performed two scans with two tools for vulnerability scanning. The first tool used was the Nessus [33] general vulnerability scanner, and the second one was the Nikto [34] Web server scanner. Both tools should generate unusual traffic, not normally seen on Web servers, but both stop short of generating attack packets that are entirely different from HTTP traffic.

Fig. 2 shows scores for packets within one hour that included a Nessus scan that lasted for some 25 minutes, from the fourth to the twenty-ninth minute. Scores for the full hour are shown to provide comparison between normal traffic and the scan. The scan traffic can be clearly distinguished from normal traffic, although some scan scores are not too high due to the previously mentioned nature of scans, i.e., that they are

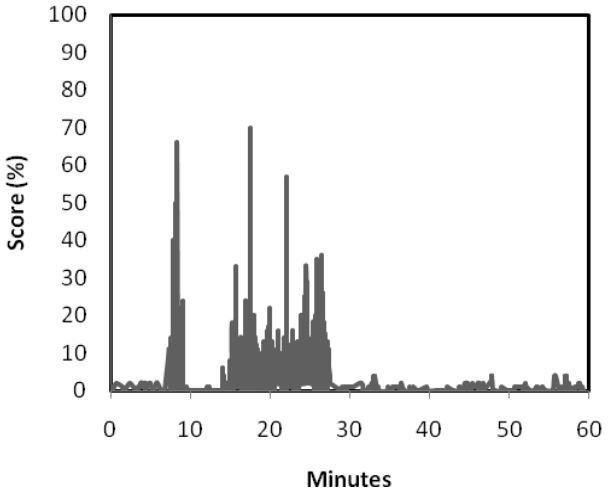

Fig. 2. Scores for one hour of network traffic that includes Nessus scan

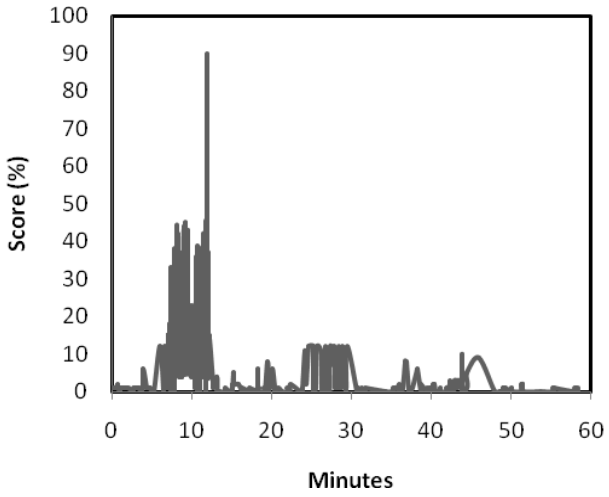

Fig. 3. Scores for one hour of network traffic that includes Nikto scan

different but still mostly HTTP traffic. Nessus, being a general vulnerability scanner, generated some HTTP packets that were not really HTTP scans and those packets scored very low.

Fig. 3 shows scores for packets within one hour that included a Nikto scan that lasted for little over six minutes, from the sixth to the twelfth minute. Similar to the Nessus scan, scores for normal and scan traffic are different enough for recognizing traffic anomaly. The scan scores are generally higher than with Nessus since Nikto is a Web vulnerability scanner that should generate more unusual HTTP traffic.

For the third test we wanted to score real attacks. For attack generation we used the Metasploit framework [35]. Metasploit provides a database of various exploits for known vulnerabilities with the possibility of using different attack payloads depending on the wanted attack result. Using various combinations of seven vulnerabilities and eight attack

TABLE I
ATTACKS WITH RELATED VULNERABILITY AND USED PAYLOAD

No.	**Vulnerability** / payload	CVE
	Apache Chunked-Encoding	2002-0392
1	adduser	
2	meterpreter-reverse_tcp	
3	shell-reverse_http	
	Apache mod_jk overflow	2007-0774
4	adduser	
5	shell-reverse_tcp	
	Apache mod_rewrite	2006-3747
6	shell-bind_tcp	
7	shell-reverse_tcp	
8	vncinject-reverse_http	
9	vncinject-reverse_tcp	
	IIS 5.0 IDQ Path Overflow	2001-0500
10	shell-reverse_http	
11	shell-reverse_tcp	
	IIS ISAPI w3who.dll	2004-1134
12	exec	
13	shell-reverse_tcp	
	Oracle 9i XDB HTTP PASS	2003-0727
14	shell-reverse_tcp	
	Xitami If_Mod_Since	2007-5067
15	shell-reverse_tcp	

payloads we generated 15 HTTP attacks over TCP port 80. Attacks with related vulnerabilities and used attack payloads are given in Table I.

Fig. 4 shows scores for 15 attacks from Table I. Different attacks generated different numbers of packets, from one to six. All attack packets were scored. The first thing worth noticing is that all packets for all attacks have scores much higher than any normal traffic packet. The lowest score for any attack packet was 55. The lowest score for the highest scoring packet in any attack was 65. All attacks can be easily detected.

This base system thus showed excellent results with normal, probe and attack traffic. The score gap between normal and attack packets is wide and provides for good detection opportunity. Setting scores above 20 to be anomalous, 100% of attacks can be detected with single digit false positives a day. If the boundary between good and bad traffic is set at 50 there would be, in this particular traffic, only one false positive in a week, but that setting might allow for some undetected attacks.

V. ADDING KEY

While experimenting with sets of delimiters, trying to find the best one, we realized that using the same method we get different sets of normal words for different set of delimiters. The previously proposed set of delimiters is based on HTTP, and it produces the biggest percentage of human language words. From the anomaly detection point of view there is no principal reason to use one set of delimiters over another. Each set of delimiters will produce a unique set of normal traffic words, byte sequences, which can be used to detect payload anomalies, words never before seen. Set of delimiters can be used as a key, which can be different for different sites or servers. Attackers can and will know the method to produce normal traffic words, but without the key, i.e., the set of delimiters, it will be very difficult, if not impossible, to craft an attack vector that will not be detected for a number of new words.

In order to test this idea we concluded several tests with different sets of delimiters, keys. Here we show results for two characteristic cases.

A. The second set of delimiters - key

In the first test we generated 20 random numbers between 9 and 127 as ASCII values to be the set of delimiters. The selected range represents printable ASCII characters as those should be major of HTTP payload transmitted values. Using this set we trained our IDS on the same set of clean traffic as we did for the base system. Then we used the system with the new set of learned words to test scores for the same set of normal traffic, probes and attacks as for the base system. We repeated the test for several random sets with similar results, of which we present one.

The set of 20 delimiters had following ASCII values

{15, 19, 20, 30, 35, 37, 38, 41, 47, 56,
58, 63, 68, 69, 90, 97, 107, 109, 114, 122}

Fig. 5 shows the dependency of the number of words learned on the number of hours of network traffic used for training. There is the same general tendency that the number of learned words stabilizes after 96 hours. The total number of learned words was almost 42000, some 30% increase. This could have been expected since, unlike the original set of delimiters optimized for HTTP traffic, these delimiters were randomly selected. The increase does not have significant influence on the size of the hash table used to store learned words or on the system performance.

Detection results with new the set of limiters and learned words were very promising. The scan of normal traffic generated scores similar to the ones with original delimiters. Most packet scores were below ten, twenty were above 20 and the odd packet from the original detection scored even lower, 47.

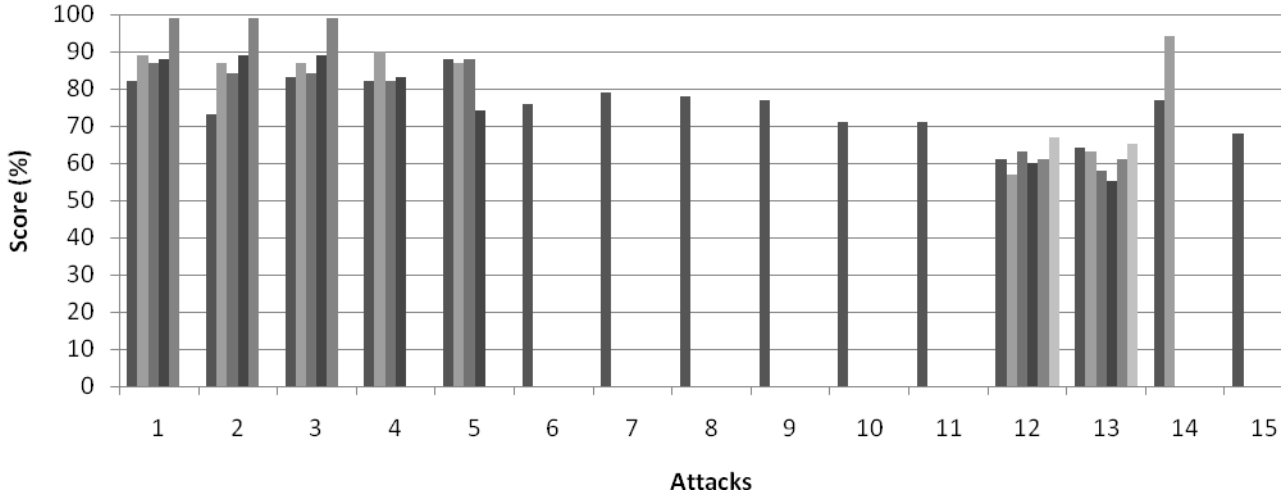

Fig. 4. Scores for each packet in 15 attacks

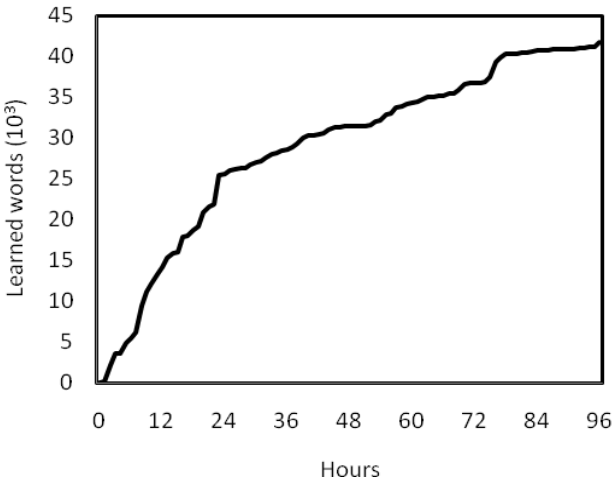

Fig. 5. Number of learned words as a function of hours of network traffic for the second set of delimiters (20 random values between 9 and 127)

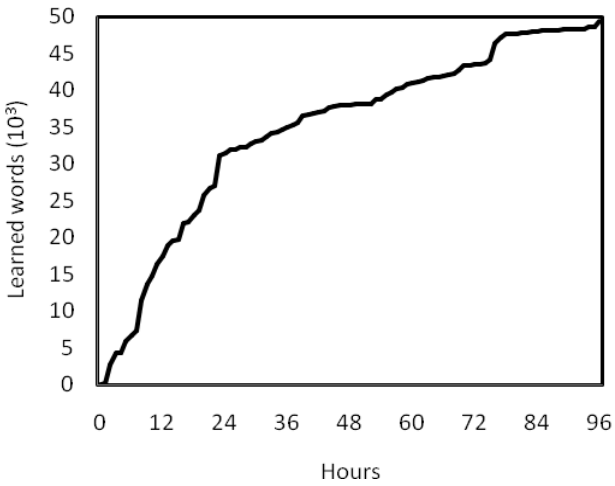

Fig. 7. Number of learned words as a function of hours of network traffic for the third set of delimiters (15 random values between 9 and 127)

Scores for Nessus and Nikto scans were very similar to scores with the original delimiters, with packets in the scan period scoring higher than normal packets. No diagram of this data is presented here, to save space, since it is almost the same as the original one.

Fig. 6 shows scores for 15 attacks from Table I using the new set of limiters and learned words. Again all packets for all attacks have scores much higher than any normal traffic packets. The lowest score for any attack packet was again 55 and that also was the lowest score for the highest scoring packet in any attack. Scores are on the average lower than the original ones, but still all attack scores were far from normal traffic scores.

Overall results with the random set of 20 delimiters, i.e., the key, support the stated idea that the base system could be used as an intrusion detection system with a site-selected key.

B. The third set of delimiters - key

In the second test we tried using a smaller number of delimiters. We generated 15 random numbers between 9 and 127 as ASCII values to be the set of delimiters. Using this set we went through the same learning and detection phase as in the first two cases. We repeated the test for several random sets with similar results, of which we present one.

The set of 15 delimiters had the following ASCII values

{ 9,20,34,36,53,60,63,64,66,69,71,97,103,111,116}

Fig. 7 shows the dependency of the number of words learned on the number of hours of network traffic used for training. There is the same general tendency that number of learned words stabilizes after 96 hours. The total number of learned words was almost 50000, approximately a 50% increase from the original set of delimiters, and a 20% increase from the second set of words. This might be result of fewer limiters providing for more possible combinations of longer words. The increase again, however does not have significant influence on the size of the hash table or on system performance.

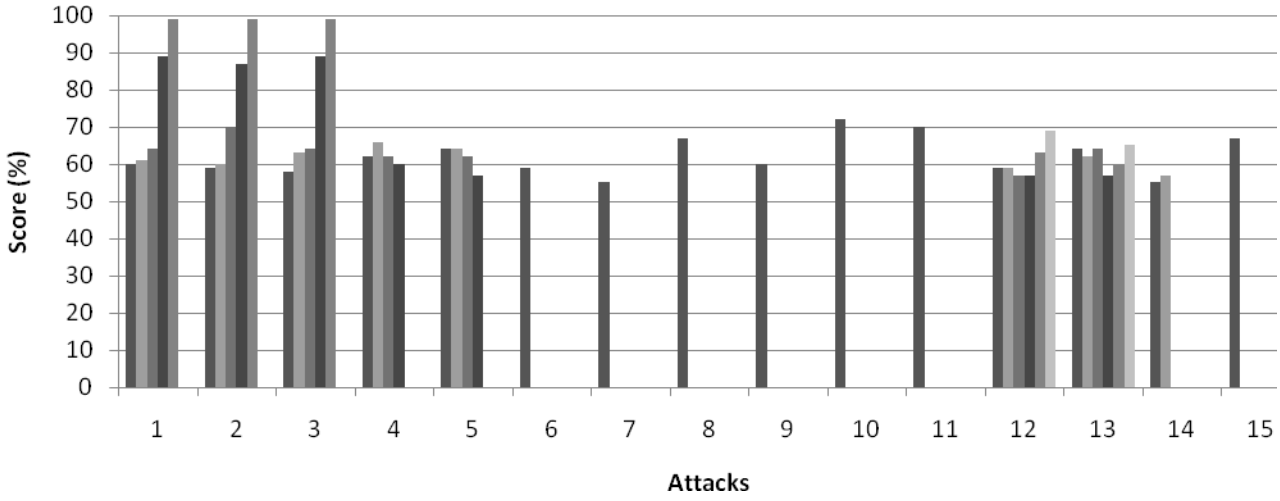

Fig. 6. Scores for each packet in 15 attacks with the second set of delimiters (20 random values between 9 and 127)

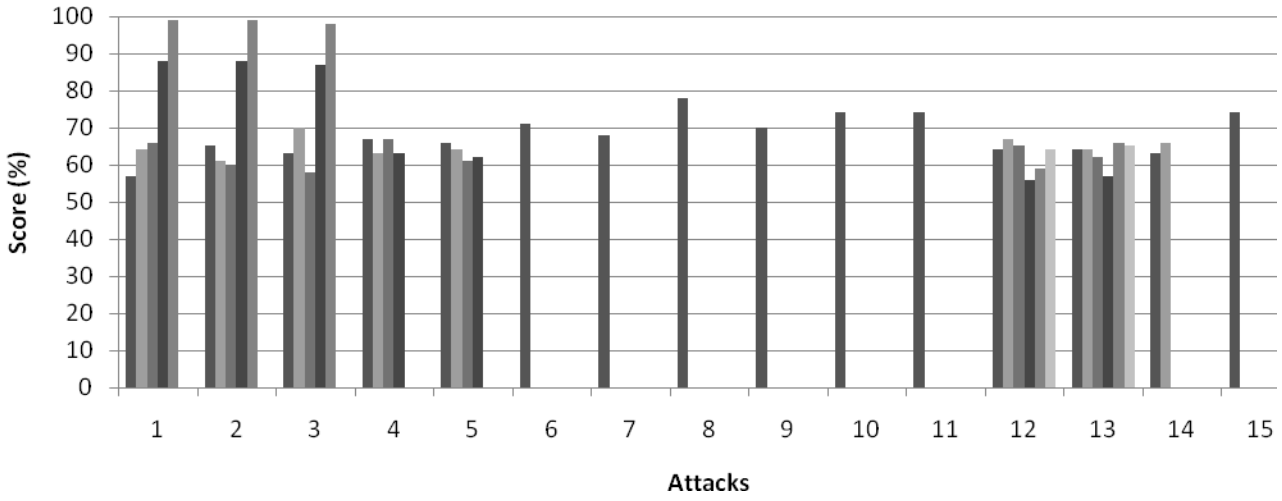

Fig. 8. Scores for each packet in 15 attacks with the third set of delimiters (15 random values between 9 and 127)

Detection results with this set of delimiters and learned words also supported the suggested approach. The scan of normal traffic generated scores similar to the ones with the original delimiters. Most packet scores were below ten, 31 were above 20, and the odd packet from the original detection scored 51.

Scores for Nessus and Nikto scans were very similar to the scores for the original delimiters, with packets in the scan period scoring higher than normal packets.

Fig. 8 shows scores for 15 attacks from Table I using the third set of limiters and learned words. This time the attack packets scored higher than with the second set. The lowest score for any attack packet was 56. The lowest score for the highest scoring packet in any attack was 66.

Again, results with a random set of 15 delimiters, as the key, showed that the base system could be used as an intrusion detection system with a site-selected key.

C. An example

We present one small and simplified example in order to show the need for and gain from a key, i.e., a site-specific set of delimiters. The example shows how an attacker might try to blend in an attack payload with normal traffic words. Knowledge of the algorithm of base system and the delimiters helps him craft a payload with as many normal words as possible. Such a payload would have a lower score and the attack might pass undetected.

Attacks number 6, 7, 8 and 9 exploit Apache mod_rewrite vulnerability with four different attack payloads. Metasploit creates an HTTP packet for this vulnerability using the following statement:

```
uri= "/#{rewritepath}/ldap://"+
rand_text_alphanumeric(rand(16))+"/"+
rand_text_alphanumeric(rand(32))+"%3f"
+rand_text_alphanumeric(rand(8))+"%3f"
+rand_text_alphanumeric(rand(8))+"%3f"
+rand_text_alphanumeric(rand(16))+"%3f"+
rand_text_alphanumeric(rand(8))+"%3f%90"
uri += payload.encoded
```

This means that there are six parts of the HTTP packet that consist of random alphanumeric stream of up to 16, 32, 8, 8, 16 and 8 characters for each part respectively. At the beginning and between the random parts are fixed byte sequences. At the end comes “%3f%90” and the encoded payload. Usually this encoded payload would increase the score for the packet and the attacker would try to use appropriate encoding to hide the attack. Since payload encoding is out of the scope of this paper we will not do any changes to this part of packet. Without loss of generality we will adjust only the beginning part of the attack vector to try to lower its score.

The original Metasploit HTTP packet payload for this attack was:

```
"GET
/1/ldap://2QeT9jN5nS4QA9/HTiYB1T8LhY9Az9DT
R9%3feG%3fu%3fHT%3fk%3f%90"+
encoded.payload
```

The encoded payload was different for four different attack payloads. Random characters (bolded) are most probably not in the set of learned words and increase the scores for this packet in addition to the scores caused by encoded payload. The scores for these four attacks using the base system were 76, 79, 78 and 77 respectively.

Knowing the delimiters and the detection algorithm we tried to manually create an HTTP packet with as many learned words as possible, effectively lowering its score. Instead of a random stream of characters we used small words we knew were learned (GET, HTTP) with the known delimiter “.” and tried to put as many of them as possible within the allowed number of characters. The resulting HTTP packet was:

```
"GET
/1/ldap://GET.GET.GET.GET/GET.GET.GET.GET.
GET.GET.GET.GET.%3f.HTTP.%3f.HTTP.%3f.GET.
GET.GET.%3f.HTTP.%3f%90"+ encoded.payload
```

It was still valid attack but its detection scores on base system for four different payloads were 50, 46, 59 and 45. These scores were significantly lower than the original ones. They might not be low enough to avoid detection, but they do prove the idea. Actual attack payload encoding might bring the score even lower.

Unlike our base system, our keyed systems with the second and the third set of delimiters were not fooled by this change of HTTP packet. Detection scores were even higher. For the second set of delimiters, scores for manually created packets were 62, 63, 68 and 68, compared with 59, 55, 67 and 60 for the original Metasploit attacks. For the third set of delimiters, scores for manually created packets were 78, 74, 81 and 84, compared with 71, 68 78 and 70 for the original Metasploit attacks. The reason for this is obvious. Since the second and the third set of delimiters were "unknown" to us we did not know how to prepare the payload to avoid detection.

Thus, the proposed keyed IDS implementation makes it extremely difficult for attackers to add malicious byte sequences to packet payloads and avoid detection. Knowledge of the algorithm being used does not help them at all, without knowledge of the particular set of delimiters used on the site they plan to attack. Without knowing the exact set of delimiters, attackers do not know how to position malicious payload parts within packet payloads.

It is important to point out that actual payload encoding is far from trivial and sometimes effectively impossible. Our manual packet creation used a very convenient vulnerability just as proof of concept and support for the proposed keyed IDS. In a general case our base system could detect most of the automated attacks.

VI. Conclusion

Applying Kerckhoffs' principle for intrusion detection could give the upper hand to the detection side, over attackers. Practical application of the principle is presented through implementation. The results show that implementation is not less good at detection and not more complicated to implement than other currently proposed anomaly detection network intrusion detection systems. The presented system can protect HTTP traffic from intrusions that try to evade anomaly detection. This protection is based on different keys, sets of delimiters, used to create sets of words seen in attack-free traffic. Each particular site can have a different key, and thus a different set of normal words.

The issue, known from cryptography, of key maintenance might need further research. A procedure for key (delimiters) changes due to compromise or time limit on use needs to be developed. It requires creating a set of normal words with a new set of delimiters. The learning process is short, requiring only minutes, and can be fully automated, but it requires keeping records of normal traffic. As a matter of fact normal traffic records should also be kept current due to its dynamic nature, and a set of normal words needs to be updated on regular basis regardless of key change. This is one of direction of our future research.

The stated idea can be generalized and applied to protocols other than HTTP, and other than text-based. Every network packet payload could be tokenized using a set of token boundaries, delimiters. Certainly, issues of selection of delimiters need further research. Similar to cryptographic keys, there might be good and weak keys, and the number of delimiters, i.e., the size of the set will impact performance and security.

References

[1] R. Chinchani, E.V.D. Berg, "A fast static analysis approach to detect exploit code inside network flows," International Symposium on Recent Advances in Intrusion Detection (RAID), 2005

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

[3] P.Fogla, et al., "Polymorphic Blending Attacks", USENIX Security, Boston, MA., 2006.

[4] C.Kruegel, T.Toth, E.Kirda, "Service specific anomaly detection for network intrusion detection", In: Proc. of ACM Symposium on Applied Computing, pp. 201-208, 2002

[5] T. Toth and C. Kruegel, "Accurate buffer overflow detection via abstract payload execution," RAID, 2002.

[6] M. Mahoney, "Network traffic anomaly detection based on packet bytes," ACM-SAC, 2003.

[7] K. Wang and S.J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," RAID, 2004.

[8] Federal Information Processing Standards, "Data Encryption Standard (DES)", Publication 46 – 3, 1999.

[9] Federal Information Processing Standards, "Advanced Encryption Standard (AES)", Publication 197, Nov 2001

[10] D. Wagner and D. Dean, "Intrusion Detection via Static Analysis," IEEE Security & Privacy, 2001.

[11] M.Rash, A.Orebaugh, G.Clark, B.Pinkard, J.Babbin, "Intrusion Prevention and Active Response : Deploying Network and Host IPS" Syngress Publishing; 1 edition, February 1, 2005

[12] SANS Institute, "SANS Top-20 2007 Security Risks (2007 Annual Update)", http://www.sans.org/top20/

[13] R.Sekar, et al., "Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions", in ACM Conference on Computer and Communications Security, Washington, D.C., 2002.

[14] R.Vargiya, P.Chan, "Boundary detection in tokenizing network application payload for anomaly detection", In: Proc. of ICDM Workshop on Data Mining for Computer Security, pp. 50-59, 2003

[15] S.Zanero, S.M.Savaresi, "Unsupervised learning techniques for an intrusion detection system", In: Proc. of ACM Symposium on Applied Computing, 2004.

[16] C.Kruegel, G.Vigna, "Anomaly detection of web-based attacks", In: Proc. of 10th ACM Conf. on Computer and Communications Security, pp. 251-261, 2003

[17] C. Kruegel, et al., "Polymorphic Worm Detection Using Structural Information of Executables", Symposium on Recent Advances in Intrusion Detection, Seattle, WA., 2005

[18] X. Wang., et al., "SigFree: A Signature-free Buffer Overflow Attack Blocker",USENIX Security, Boston, MA., 2006.

[19] P. Akritidis, E. P. Markatos, M. Polychronakis, and K. D. Anagnostakis, "Stride: Polymorphic sled detection through instruction sequence analysis", In Proceedings of the 20th IFIP International Information Security Conference (IFIP/SEC 2005), 2005.

[20] K. Wang, G.Cretu, S.Stolfo, "Anomalous payload-based worm detection and signature generation", In: Recent Advances in Intrusion Detection (RAID), 2005

[21] K. Wang, J. Parekh, S. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," RAID, 2006

[22] K. Rieck, P. Laskov, "Language Models for Detection of Unknown Attacks in Network Traffic," Journal in Computer Virology, Vol. 2, No. 4. , pp. 243-256., February 2007

[23] D. Wagner and P. Soto, "Mimicry Attacks on Host-Based Intrusion Detection Systems," ACM CCS. 2002.
[24] K.M.C. Tan, K.S. Killourhy, and R.A. Maxion, "Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits," RAID, 2002.
[25] O. Kolesnikov, D. Dagon, and W. Lee, "Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic," Technical Report GIT-CC-04-13, College of Computing, Georgia Tech, 2004.
[26] P. Fogla P, W. Lee, "Evading network anomaly detection systems: formal reasoning and practical techniques," ACM Conference on Computer and Communications Security (CCS), 2006
[27] A. Kerckhoffs, "La cryptographie militaire," Journal des sciences militaires, vol. IX, pp. 5–83, Jan. 1883, pp. 161–191, Feb. 1883. (http://petitcolas.net/fabien/kerckhoffs/)
[28] C. Shannon, "Communication theory of secrecy systems," Bell Systems Techn. Journal, 28:656-715, 1949.
[29] J.D. Saltzer, M.D. Schroeder, “The Protection of Information in Computer Systems,” in Proceedings of the IEEE, v 63 no 9, pp 1278–1308., 1975.
[30] M. Barreno, et al. "Can Machine Learning Be Secure?" ASIACCS, 2006.
[31] M. Roesch, "Snort: Lightweight intrusion detection for networks", USENIX Large Installation System Administration Conference LISA, pp. 229-238, 1999
[32] SourceFire Inc. Snort rules. 2008 [cited 2008 Feb 28]; Available from: http://www.snort.org/pub-bin/downloads.cgi
[33] The Nessus vulnerability scanner, Tenable Network Security, available online, http://www.nessus.org
[34] Web server scanner, available online, http://www.cirt.net/code/nikto.shtml
[35] The Metasploit project, available online, http://www.metasploit.com