NIDS Based on Payload Word Frequencies and Anomaly of Transitions

Sasa Mrdovic
University of Sarajevo
Faculty of Electrical Engineering
sasa.mrdovic@etf.unsa.ba

Branislava Perunicic
University of Sarajevo
Faculty of Electrical Engineering
brana_p@hotmail.com

Abstract

This paper presents a novel payload analysis method. Consecutive bytes are separated by boundary symbols and defined as words. The frequencies of word appearance and word to word transitions are used to build a model of normal behavior. A simple anomaly score calculation is designed for fast attack detection. The method was tested using real traffic and recent attacks to demonstrate that it can be used in IDS. Tolerance to small number of attack in training data is shown.

1. Introduction

The intrusion detection is a standard part of the system for the information system protection. It is designed to detect security policy violations. In a perfect world preventive controls would stop such violations, but so far this is not a realistic assumption [1]. Since 1980s, intrusion detection systems (IDS) have evolved following changes in environment they operate in. Most of the recent network attacks are aimed to the application level [2] [3] because most of the lately found vulnerabilities appear in networked applications [4] [5]. Moreover, most of the attacks at lower protocol levels may now be prevented with well established network protection tools as firewall. Thus, the current network IDS as a rule analyze network packet payload.

A properly configured signature based IDS can detect all known attacks with very little false alarms, if any. Such IDS also aims to detect variations of known attacks. However, that effort might be futile [6]. With the present rate of 10 and more new vulnerabilities discovered every day [7] new attacks without a signature cannot be detected by a signature based IDS. The anomaly based IDS, however, should be able to detect unknown attacks. Although the complete anomaly detection paradigm has been recently challenged [8], it still remains the only realistic option for these, so called, zero-day attacks.

This paper presents a novel method for network intrusion detection based on anomalies found in the network packet payload. The payload is divided into multi-byte sequences defined as words. These words are used to model normal behavior. The model is built using the frequency of occurrence of words and the frequencies of one word following another. The obtained evaluation results seem very promising.

The following section describes related work. The Section 3 introduces the idea and the method based on it. The method implementation is presented in the Section 4. The evaluation results are provided in Section 5. Finally, concluding remarks are given, and possible future work is discussed in Section 6.

2. Related work

Network intrusion detection systems that analyze packet payload anomalies are focus of an intensive current research. Various approaches have been proposed. Specific knowledge on applications that offer network services is used in [9], where type, length, and character frequencies of the request are analyzed. Frequencies of first 48 bytes for nine most used protocols are basis for the model in [10]. Combination of signature and anomaly based detection using state-machine for analysis of a small payload portion is topic of [11]. Two-tier architecture is proposed in [12]. The first tier calculates one byte value from packet payload. The second tier uses self-organizing maps to classify packets using both this value and packed header data. Packet payload byte frequencies are the main constituent of the model in [13]. This approach is improved in [14]. Syntax and semantic information from packet payload are used in [15] and [16]. Detection of executable code bytes in packet payload is considered in [17] and [18]. Analysis of payload based application level network anomaly detection with some new ideas for improvements is focus of [19]. Based on suggestions from [20], [21] explains how to create attacks that methods suggested in [9], [22], [10] and [13] cannot detect. Authors claim that byte frequency based anomaly IDS are open to attacks and may be easily evaded. Inspection and analysis of network traffic using three levels of granularities, traffic flow, packet header, and payload is subject of [23].

The division of payload by using delimiters was first suggested in [24], but the main issue of that paper was the choice of the boundary characters, delimiters, not analysis of payload parts. The frequency analysis of multiple consecutive bytes, n-grams, in payload has been proposed in [25]. The use of payload words, seen as consecutive bytes separated by delimiters, to make a language model was considered in [26]. Authors of [27] take a different view and use HTTP GET request parameters and their values as the starting point for the model: the length, character positions, the structure, the values and existence of parameters are considered. This idea is further developed and improved in later papers [28] [29]. Yet another approach to analysis of HTTP request parameters is taken by [30], where Deterministic Finite Automata induction algorithm is used to detect malicious requests.

3. Method

The method proposed here uses words to model normal behavior. Words are defined as groups of consecutive bytes in the network packet payload delimited by boundary symbols. The meaningful words are defined as key words from the used protocol or as words in the language of the transmitted message. If proper boundary symbols are used as delimiters, the percentage of meaningful words might be sizeable. Most of the text based protocols use similar sets of boundary symbols, as required by their semantics.

The testing of proposed approach uses HTTP. The majority of effective malicious activity has become Web-based [2]. Here are some of the reasons. The standard HTTP TCP port 80 is almost always open for outgoing and incoming traffic on firewalls. Web application vulnerabilities account for almost a half of all vulnerabilities discovered in the past year [31]. In addition, the much used e-mail service is now normally offered through Webmail.

Based on the results from [24], and [26], and our own experiments, a set of 20 boundary symbols that provides the highest percentage of meaningful words for HTTP is defined. These 20 delimiters are:

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

The above set of boundary symbols may result in any length of a word. In order to keep the number of words down, and get a smaller model, allowed word length is limited to the range from 3 to 16. Words shorter than 3 bytes are ignored, since they hardly may be an important part of an attack. Even if there is no boundary symbol the word ends after 16 consecutive bytes and a new word begins. This word length limit was successfully applied in [26].

The next issue is how to build a model of normal behavior. Recent comparisons of anomaly detection techniques for HTTP [32] imply the following conclusion: since normal requests have a meaning, if an algorithm can get the sense of a HTTP request, it improves its ability to discriminate between a normal and an abnormal message. A conversation in some language or an HTTP request-reply must have an implication and should be easily set apart from a conversation in some other language or gibberish. The words in a language have a probability of appearance, and they follow some rules of precedence Therefore, to build a model, word frequencies and word transitions may be used. Word transitions are probabilities that certain word would come after some other word. Those transitions are a part of the Markov model proposed in the first IDS paper [1]. It may be expected that malicious requests and attacks would have both a significantly different distribution of word frequencies and considerably different distribution of word transitions.

4. Implementation

To test hypothesis stated in previous section an application implementing the proposed method was created and tested. It should be noted that IDS testing is still an open issue. In particular, the HTTP intrusion detection has its specific difficulties. The current testing methods problems are point out in [32]. The best known and most widely used data sets for IDS testing were proposed by DARPA/MIT Lincoln Laboratories in 1998 and 1999 [33] [34]. The choice of these data sets was questioned soon after their publication [35] [36]. Two main reasons why DARPA data sets are not adequate for current HTTP IDS testing may be summarized as follows: To begin with, both traffic and even more attacks in those data sets are obsolete. Next, there are only four web attacks in the data sets. Therefore in this paper EE department of the Sarajevo University traffic was used for testing as a typical traffic.

4.1. Learning

During the learning phase a model of normal, which is attack free, traffic is built. There are number of issues with machine learning [37] but it is out of scope of this paper. One important issue is the availability of a really clean traffic [8]. To deal with this problem, the proposed method is created to be rather tolerant to few attacks in training data.

The supervised learning was used for the making of the model of the normal behavior. The 96 hours of recorded traffic from the EE department Web server was cleaned using a combination of signature based network intrusion detection Snort [38] fully tuned, with all rules activated, and supplied with latest signature content of Snort rules. Lastly, the manual inspection was applied. Such training data were supplied to the system as an apparently attack free traffic.

All incoming traffic packet payloads were scanned and divided into words using set of 20 previously defined delimiters. All words found in normal traffic were stored in a hash table, together with the frequencies with which they appear. Number of learned words leveled after the 96 hours of learning as seen in the Fig, 1. Total number of learned normal words was in the region of 33 000.

A huge 33 000 x 33 000 matrix is needed to store all transitions from one word to another. Since hash table showed a big diversity of numbers of word appearances, a smaller matrix was likely to meet requirements. Consequently, the transition matrix was made using only the words that appear more than ten times, since these words comprise 10% of all words. Word transitions not found this in matrix are considered as rare and a high anomaly score would be assigned to them.

4.2. Detection

To test the ability to detect attacks, all Web server incoming traffic payloads were scanned and divided into words as described above. Each packet payload was first scored based on the learned frequencies of the words in packet using the following formula:

$$S_w = \frac{\sum_{i=1}^{k} \frac{1}{n(w_i)}}{k} \qquad (1)$$

In this formula k is the number of words in a payload and $n(w_i)$ is the number of appearances of the word w_i in learned model. For the words that were not seen in normal traffic thus having $n(w_i) = 0$, the corresponding term in the sum is set to 2. In this way the words that seldom appear in normal traffic would make the score higher. On the other hand, words that have a high frequency of appearance in normal traffic would make almost no contribution to the sum. This prevents malicious payloads, padded with frequent words to mask attack vector, to have a low score. Besides, even if the training traffic is not absolutely clean of attacks, these attacks would still have a score higher than normal traffic, assuming that the number of such attacks in training data was low. This assumption is more realistic than the standard assumption that there is an attack free traffic available for learning. Test with attacks in training data will be presented later.

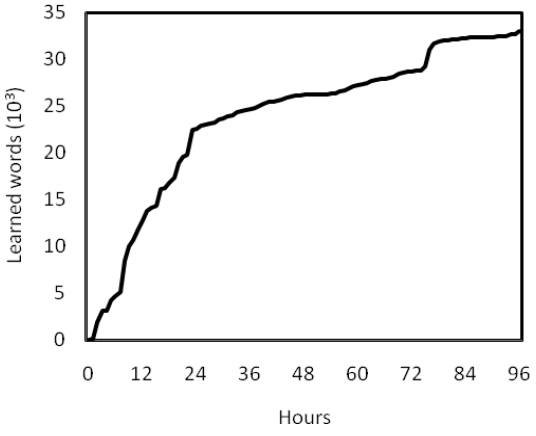

Fig. 1. Number of learned words as a function of hours of traffic

First test was with one hour of a traffic containing a vulnerability scan performed using Nikto, Web server scanner. Although there is no real attack, such scan is not a usual traffic on Web servers. The Fig. 2 shows word scores for all packets within one hour that included Nikto scan that lasted from 6th to 12th minute. Scores for full hour are shown to provide comparison between normal traffic and scan. The scan traffic can be clearly distinguished from the normal one, although some of the packets in the normal traffic had higher scores than the others. This issue will be addressed later. The results confirm prediction that an abnormal traffic, as a scan, has different distribution of word frequencies than a normal traffic.

Next, each packet payload was scored based on the learned frequencies of the word transitions in packet using formula similar to the one for words:

$$S_t = \frac{\sum_{i=1}^{m} \frac{1}{n(t_i)}}{m} \qquad (2)$$

Here m is the number of word transitions in a payload and $n(t_i)$ is the number of times transition t_i occurred during training. Similarly to word score, transitions that are rare in normal traffic would make the score higher. For the transitions that were not seen in normal traffic thus having $n(t_i) = 0$, term of sum is again set to 2. Frequent transitions would have very little effect on total transition score.

The importance of putting into service word transition in the model of the normal behavior may be explained in the following way: The stuffing of frequent words into the attack packet to lower its word score, and so mask the attack, might be

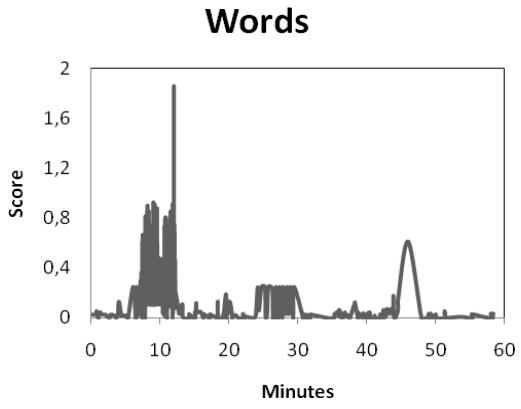

Fig. 2. Word scores for one hour of traffic that includes Nikto scan

difficult, but it is still possible. Doing something similar with word transitions is also possible, but without doubt much harder to do.

The same hour of traffic with Nikto scan was scored with transition score. Fig. 3 shows transition scores for all packets within that hour. At first look the scores are similar to the word scores. But, a careful analysis of the numbers showed that the scan packet transition scores were generally higher and the normal traffic transition scores generally lower than the words scores in the same traffic. This can be seen on the Fig. 3. Another important fact is that if a normal packet had a higher than normal score it was usually either higher word score or a higher transition score, only rarely both. Scan packets, on the other hand, had both scores high. This is a very convenient property that suggests how to combine these scores to get an indicative total score. Since both scores have to be high for a packet to be considered abnormal, the multiplication of word and transition score is a logical choice. So for the total score the following simple formula is used:

$$S = S_w * S_t \qquad (3)$$

The Fig. 4 shows the total scores for the same hour of traffic with Nikto scan. The results are much better than for either of single scores, as predicted. The scan traffic is scored significantly different from the normal one. The scores for the normal packets are very low with very few minor exceptions. These exceptions still have much lower total score than the scan packets.

For the better visualization purposes only a half of the theoretical scale was shown. A real detection uses scores, not their visual representation and only thing that matters is that malicious packets have significantly higher scores.

A similar test with a scan using general vulnerability scanner Nessus, was performed with similar results. The results are not shown here for the lack of space.

Note that for a faster calculation of both scores inverse values of $n(w_i)$ and $n(t_i)$ are calculated and

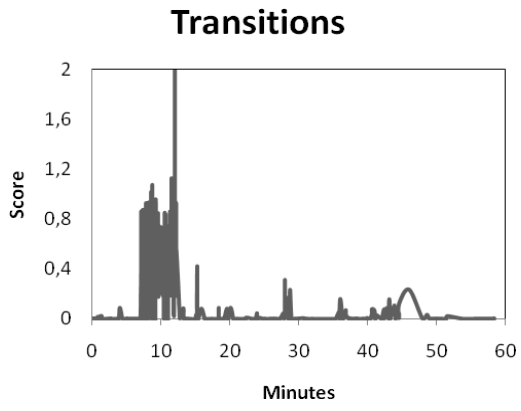

Fig. 3. Transition scores for one hour of traffic that includes Nikto scan

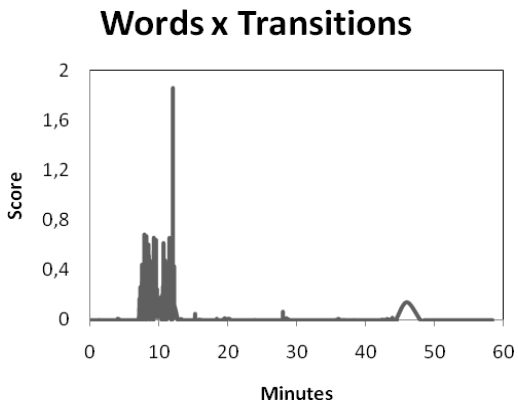

Fig. 4. Total scores for one hour of traffic that includes Nikto scan

stored before the detection phase.

5. Evaluation

The first evaluation test was performed using Metasploit framework for creation of real attacks. Metasploit has a database of various exploits for known vulnerabilities. It is possible to use different attack payloads depending on wanted attack result. The attacks in this test were made using various combinations of seven vulnerabilities and seven attack payloads. In this way eleven HTTP attacks over TCP port 80 were made. The particulars of each attack are given in the Table I.

These 11 attacks were inserted in normal traffic during one hour. Every five minutes one of the attacks was inserted. The Fig. 5 shows scores for that hour of traffic. The moment of the attack inception is very clearly seen in the graph. Since each packet was individually scored and each attack session has different number of packets, peaks have different widths. The lowest score of any attack packet was

TABLE I
ATTACKS WITH RELATED VULNERABILITY AND USED PAYLOAD

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

1.15.

The number of attacks was then expanded to 18 and more combinations of vulnerabilities and payloads were used for their construction. Total number of attack packets was 200 and all of them had scores above 1.15. The next test was carried out with normal traffic. Six days of traffic from the department Web server were scored. The aim was to get the number of false positive alarms. Threshold for an anomalous score was varied in the range from 0.2 to 2.0. Receiver operating characteristics (ROC) curve, which is usual tool for reporting accuracy of IDS results, for the system is shown on Fig. 6. False positive rate scale goes from 0 to 0.005 to provide enough detail in the part of the picture where ROC changes. With threshold of 1.0 the system detection rate is 100% with 12 false alarms a day. Results are promising and seem better than most results reported by authors mentioned in related work section. Real comparison is difficult due to reasons explained in [32].

As it was previously mentioned proposed method is tolerant to attacks in training data to some extent. It measures how unusual incoming HTTP packet is. A single attack that was a part of training data should still be unusual compared to normal traffic that makes majority of other learning data. Tests to check this feature of the algorithm were created for three attacks, number 2, 3 and 8 from Table I. Each of the attacks was first included in training data and then scored using the proposed approach. Results are given in Table II. Scores for these attacks, after they have been added to training data, have halved at least. This is not good, but it may be expected. On the other hand all packet scores, except for three, were still much higher than scores for normal packets. Although last one or two last packets of these attacks might have very low score, there are enough other packets in attack with scores high enough for detection. Consequently, even if training data is not 100% attack free system should be able to detect attacks, as long as number of attacks in training data is small enough.

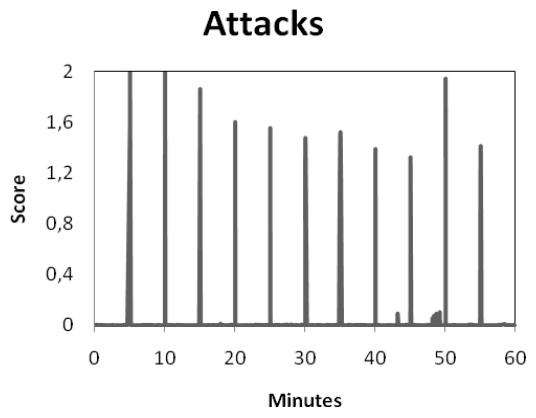

Fig. 5. Total scores for one hour of traffic that includes 11 attacks

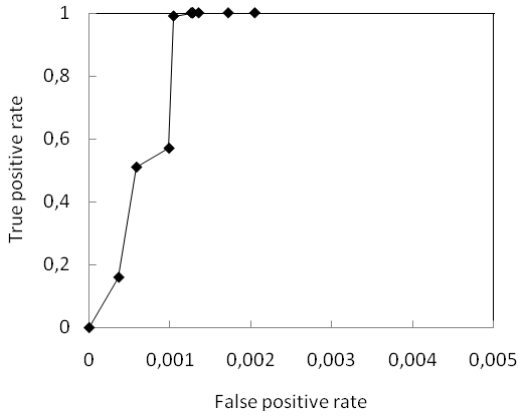

Fig. 6. ROC curve

6. Conclusion and future work

This paper presents a novel method for detection of attacks inserted in network packet payload. The method is simple, yet showed as successful in detection of all attacks it was tested with. False positive rate and number of false alarms per day are very low. The method is resistant to the padding of attacks with normal traffic patterns. Small number of attacks in training data is tolerable and does not prevent detection of these attacks.

Implementation is tested with HTTP traffic that we found to be currently the most in need to be protected. The method could work with other text based protocols like SMTP and FTP. This should be tested and is one of directions for future work. Further testing with different data should be performed to confirm the presented results. Using different set of boundary symbols, for each implementation, suggested in [39], should provide an additional resistance to the detection evasion.

TABLE II
ATTACKS SCORES COMPARISON FOR (UN)CLEAN TRAINING DATA

Attack no. in Table I	Not in training data	In training data
2	1,697458	0,568382
2	1,788083	0,900336
2	1,745897	0,877035
2	1,803798	0,004466
2	1,987421	0,000016
3	1,736318	0,872588
3	1,864706	0,936275
3	1,761585	0,885549
3	1,728683	0,000946
8	1,720629	0,484422
8	2,392497	0,602977
8	2,574378	0,64711
8	2,5483	0,639208
8	2,609535	0,655562
8	1,661021	0,416667

References

[1] D.E. Denning, “An intrusion-detection model.,” IEEE Transactions on Software Engineering, vol. 13, 1987, pp. 222-232.

[2] Internet Security Threat Report, Volume XII, Symantec Corporation, 2008.

[3] 2008 INTERNET SECURITY TRENDS, IronPort and Cisco, 2008.

[4] “CVE - Common Vulnerabilities and Exposures (CVE)”; http://cve.mitre.org/.

[5] “SecurityFocus - Vulnerabilities”; http://www.securityfocus.com/vulnerabilities.

[6] Y. Song et al., “On the infeasibility of modeling polymorphic shellcode,” 14th ACM conference on Computer and communications security, 2007, pp. 541-551.

[7] US –National Institute of Standards and Technology, “National Vulnerability Database Home”; http://nvd.nist.gov/.

[8] C. Gates and C. Taylor, “Challenging the anomaly detection paradigm: a provocative discussion,” 2006 workshop on New security paradigms, 2006, pp. 21-29.

[9] C. Krügel, T. Toth, and E. Kirda, “Service specific anomaly detection for network intrusion detection,” 2002 ACM symposium on Applied computing, 2002, pp. 201-208.

[10] M.V. Mahoney, “Network traffic anomaly detection based on packet bytes,” 2003 ACM symposium on Applied computing, 2003, pp. 346-350.

[11] R. Sekar et al., “Specification-based anomaly detection: a new approach for detecting network intrusions,” 9th ACM conference on Computer and communications security, 2002, pp. 265-274.

[12] S. Zanero and S.M. Savaresi, “Unsupervised learning techniques for an intrusion detection system,” 2004 ACM symposium on Applied computing, 2004, pp. 412-419.

[13] K. Wang and S.J. Stolfo, “Anomalous Payload-Based Network Intrusion Detection,” 7th International Symposium on Recent Advances in Intrusion Detection (RAID), 2004.

[14] K. Wang, G. Cretu, and S.J. Stolfo, “Anomalous Payload-Based Worm Detection and Signature Generation,” 8th International Symposium on Recent Advances in Intrusion Detection (RAID), 2005.

[15] R. Chinchani and E. van den Berg, “A Fast Static Analysis Approach to Detect Exploit Code Inside Network Flows,” 8th International Symposium on Recent Advances in Intrusion Detection (RAID), 2005.

[16] P. Akritidis et al., “Stride: Polymorphic sled detection through instruction sequence analysis,” 20th IFIP International Information Security Conference (IFIP/SEC 2005), 2005.

[17] C. Kruegel et al., “Polymorphic Worm Detection Using Structural Information of Executables,” 8th International Symposium on Recent Advances in Intrusion Detection (RAID), 2005.

[18] X. Wang et al., “SigFree: a signature-free buffer overflow attack blocker,” 15th conference on USENIX Security Symposium - Volume 15, 2006, p. 16.

[19] L. Zhang and G.B. White, “Analysis of Payload Based Application level Network Anomaly Detection,” 40th Hawaii International Conference on System Sciences, IEEE Computer Society, 2007, p. 99.

[20] O. Kolesnikov and W. Lee, Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic, College of Computing, Georgia Tech, 2005.

[21] P. Fogla et al., “Polymorphic blending attacks,” 15th conference on USENIX Security Symposium - Volume 15, 2006, p. 17.

[22] T. Toth and C. Kruegel, “Accurate Buffer Overflow Detection via Abstract Pay load Execution,” Recent Advances in Intrusion Detection (RAID), 2002, pp. 274-291.

[23] Y. Al-Nashif et al., “Multi-Level Intrusion Detection System (ML-IDS),” Autonomic Computing, 2008. ICAC '08. International Conference on, 2008, pp. 131-140.

[24] R. Vargiya and P. Chan, “Boundary Detection in Tokenizing Network Application Payload for Anomaly Detection,” Workshop on Data Mining for Computer Security, 2003.

[25] K. Wang, J. Parekh, and S. Stolfo, “Anagram: A Content Anomaly Detector Resistant to Mimicry Attack,” Recent Advances in Intrusion Detection (RAID), 2006, pp. 226-248.

[26] K. Rieck and P. Laskov, “Language models for detection of unknown attacks in network traffic,” Journal in Computer Virology, vol. 2, 2007, pp. 243-256.

[27] C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” 10th ACM conference on Computer and communications security, 2003, pp. 251-261.

[28] C. Kruegel, G. Vigna, and W. Robertson, “A multi-model approach to the detection of web-based attacks,” Computer Networks, vol. 48, Aug. 2005, pp. 717-738.

[29] W. Robertson et al., “Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks,” 13th Symposium on Network and Distributed System Security (NDSS), 2006.

[30] K. Ingham et al., “Learning DFA representations of HTTP for protecting web applications,” Computer Networks, vol. 51, Apr. 2007, pp. 1239-1255.

[31] SANS Institute, “SANS Top-20 2007 Security Risks (2007 Annual Update)”; http://www.sans.org/top20/.

[32] K. Ingham and H. Inoue, „Comparing Anomaly Detection Techniques for HTTP“, 2007 Conference on Recent Advances in Intrusion Detection (RAID), 2007.

[33] R. Lippmann et al., “Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation,” DARPA Information Survivability Conference and Exposition, 2000, pp. 12-26 vol.2.

[34] L. Richard et al., “The 1999 DARPA off-line intrusion detection evaluation,” Computer Networks, vol. 34, Oct. 2000, pp. 579-595.

[35] J. McHugh, “The 1998 Lincoln Lab IDS Evaluation—A Critique,” 3rd International Workshop on Recent Advances in Intrusion Detection (RAID), 2000.

[36] J. McHugh, “Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” ACM Trans. Inf. Syst. Secur., vol. 3, 2000, pp. 262-294.

[37] M. Barreno et al., “Can machine learning be secure?,” 2006 ACM Symposium on Information, computer and communications security, 2006, pp. 16-25.

[38] M. Roesch, “Snort-Lightweight Intrusion Detection for Networks,” 1999 USENIX LISA Systems Administration Conference, 1999, pp. 229–238.

[39] S. Mrdovic and B. Perunicic, “Kerckhoffs’ Principle for Intrusion Detection”, Networks 2008, (to be published)