Detecting Unknown Network Attacks Using Language Models

Konrad Rieck and Pavel Laskov

Fraunhofer-FIRST.IDA
Kekuléstr. 7, 12489 Berlin, Germany
{konrad.rieck, pavel.laskov}@first.fraunhofer.de

Abstract. We propose a method for network intrusion detection based on language models such as n-grams and words. Our method proceeds by extracting these models from TCP connection payloads and applying unsupervised anomaly detection. The essential part of our approach is linear-time computation of similarity measures between language models stored in trie data structures.

Results of our experiments conducted on two datasets of network traffic demonstrate the importance of higher-order n-grams for detection of unknown network attacks. Our method is also suitable for language models based on words, which are more amenable in practical security applications. An implementation of our system achieved detection accuracy of over 80% with no false positives on instances of recent attacks in HTTP, FTP and SMTP traffic.

1 Introduction

Detection of unknown attacks is a long-standing issue on a wish-list of security practitioners. While it is often claimed that current applications and infrastructures for tracking vulnerabilities and their exploits provide adequate protection by means of attack signatures, there exist numerous examples of previously unknown attacks, notably worms (e.g. 1) and zero-day exploits (e.g. 2), that have defeated signature-based defenses. Furthermore, it often does not suffice for a signature to be available – deployed signatures must be kept up-to-date by security administrators.

Discussion about unknown attacks has been carried out in various parts of the intrusion detection community. For misuse detection, it centers around the issues of making signatures more generic – and capable of at least not to be fooled by mutations of known attacks (3; 4; 5; 6; 7; 8). There is, however, a growing consensus that genuinely novel attacks can only be detected by anomaly detection tools, at a cost of having to deal with false positives which may also be valid anomalies.

A large amount of previous work has been done on anomaly detection in network traffic (e.g. 9; 10; 11; 12; 13; 14). The main hurdle on the way to its acceptance in practice is a high rate of false positives. Most of the previous

R. Büschkes and P. Laskov (Eds.): DIMVA 2006, LNCS 4064, pp. 74–90, 2006.

approaches do not deliver sufficient accuracy in an acceptable range of false-positive rates. Hence further improvements of anomaly-based intrusion detection techniques are highly desirable.

Apart from algorithmic differences, the main issue underlying anomaly detection approaches is the features they operate on. Some early approaches consider only packet header information or statistical properties of sets of packets and connections (13; 15). This information has proved to be useful for detection of certain kinds of malicious activity such as probes and port scans, yet it usually does not suffice to detect attacks that exploit semantic vulnerabilities of application-layer protocols and their implementations.

Recently, techniques of anomaly-based network intrusion detection have been proposed that analyze packet and connection payloads (11; 16; 17; 18; 10; 9). These techniques proceed by defining features over payloads and deriving models of normality based on these features. Packets and connections that do not fit into such models are considered anomalous and trigger alarms. All of these methods make use of relatively simple features computed over payload bytes.

The main thesis of this contribution is that further improvement of detection accuracy can be achieved by more advanced features defined over *byte sequences*. The reason why byte sequences may be more successful in description of features indicative of malicious content can be seen by comparing network protocols and natural languages. The content of both is characterized by rich syntax and semantics, and discrimination between different categories is only possible in terms of syntactic and semantic constructs. For both network protocols and natural languages, extensive effort has been made to describe important concepts in terms of rules, only to find out that rules can hardly encompass the full generality of underlying content. Protocols and natural languages possess grammatical structure and yet recovery of this structure is stymied by uncertainty and ambiguity. In view of the linguistic analogy, one can see that detection of misuse and anomalous patterns amounts to learning syntactic and semantic fragments of an underlying protocol language. Hence it is clearly promising to apply the machinery of natural language processing to network intrusion detection.

Byte sequences can be represented by so-called *n-grams*, sequences of n consecutive symbols. Such representations have been previously used to model traces of system calls (e.g. 19; 20; 21; 22; 23; 24), but surprisingly have not been applied in the context of network intrusion detection for $n > 1$. The main technical difficulty that needs to be addressed for analysis of byte sequences is:

> How can language models of packet and connection payloads, such as n-grams, be efficiently extracted and compared?

Having efficient techniques for comparison of language models, one can apply *unsupervised anomaly detection algorithms* to identify unusual events. Hence, we focus our attention on methods for computing similarity measures between such models. To address this problem we propose (a) a representation of n-grams using *tries* and (b) a novel method for comparison of tries in linear time.

2 N-Grams, Tries and Anomaly Detection

2.1 N-Grams of TCP Connections

To motivate the subsequent presentation of our method, we begin with an examples that illustrates the utility of language models for discrimination of network attacks and normal data. Fig. 1 shows the differences between 3-gram frequencies of an IIS unicode attack and normal HTTP traffic. Due to the large space of possible 3-grams the plot is limited to 3-grams present in the IIS unicode attack.

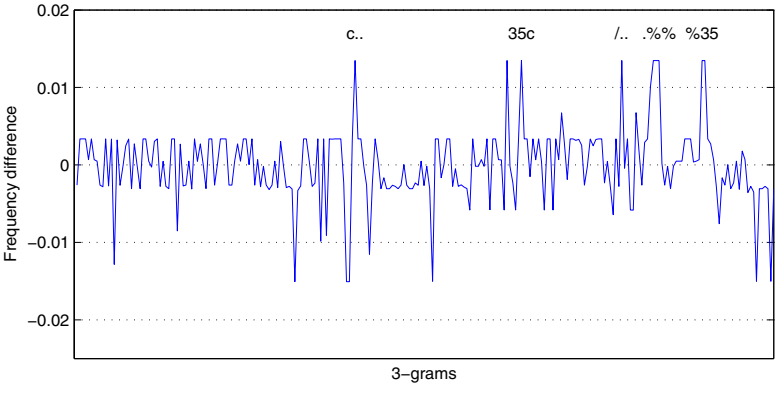


Fig. 1. Frequency differences of 3-grams for an IIS unicode attack

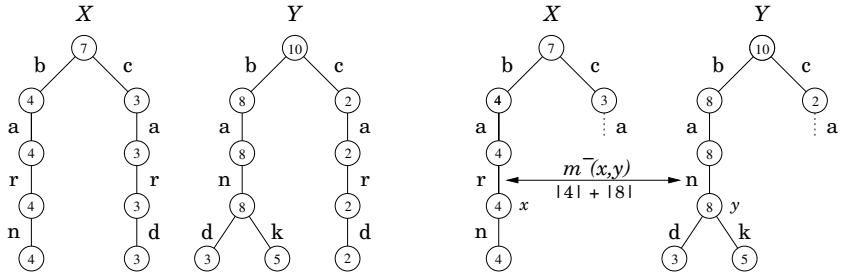
Several positive peaks in the plot, which indicate a strong deviation from normal traffic, correspond to typical 3-grams of the attack, e.g. "35c", "/.." and "%35". These 3-grams manifest an essential pattern of the unicode attack "%%35c" which is converted by a vulnerable IIS server to "%5c" (ASCII code 0x35 corresponds to "5") and finally interpreted as backslash (ASCII code 0x5c). The corresponding fragment of the attack is shown below.

```
GET /scripts/..%%35c../..%%35c../..%%35c../..%%35c../..%%35c../..%%35c..
    /winnt/system/cmd.exe?/c+dir+c:
```

Although the presented example gives evidence that n-grams convey valuable information for identification of attacks, one should abstain from attempting to use n-gram frequencies in ad-hoc detection rules. Manifestation of attacks in n-grams can significantly vary, therefore a more formal approach based on measuring similarity between language models is advocated here.

2.2 Comparison of N-Grams

In order to apply anomaly detection on language models, a set of similarity measures must be provided. A large variety of such measures, which differ in the way



(a) N-gram tries (b) Comparison of n-gram tries

Fig. 2. Trie data structures (a) and their comparison (b)

they emphasize discriminative aspects of features, is available for vectorial data. We now address the problem of extending such measures to language models.

The classical scheme for storing and comparing n-gram models utilizes a hash table (e.g. 25). The n-grams extracted from a character stream and their frequencies are stored in the bins of a hash table. Assuming the size of a hash table is fixed at M, it takes on average $\Theta(M)$ to compare two hash tables containing n-grams: one needs to loop over all M bins, checking for matching and mismatching n-grams. To avoid possible hash collisions, a high value of M must be chosen in advance, which is the main computational drawback of the hash table approach.

A better alternative for storing and comparing n-grams is a *trie* data structure (26; 27). A trie is essentially an N-ary tree, whose nodes are N-place vectors with components corresponding to the characters of an alphabet of size N (28). Fig. 2(a) shows two tries X and Y containing the 4-grams {"barn", "card"} and {"bank", "band", "card"}. Each node x of the trie is augmented to carry a counter x_c reflecting the occurrences of the inserted sequence. For example the left trie in Fig. 2(a) holds 4 "barn"s and 3 "card"s.

Comparison of two tries can be carried out by enumerating matching and mismatching n-grams. Starting at the root nodes, one traverses both tries in parallel, processing matching and mismatching nodes. As an invariant, the nodes under consideration in both tries remain at the same depth, and thus the worst-case run time is $O(nk)$ for k stored n-grams.

A similarity measure over two tries X and Y can now be expressed by defining a traversal operator $\bigoplus$ and a match function m for matching and mismatching nodes x and y:

$$d(X,Y) = \bigoplus_{x,y}^{\mathrm{Trie}X,Y} m(x,y), \text{ where } m(x,y) = \begin{cases} m^+(x,y) & \text{if } x = y \\ m^-(x,y) & \text{otherwise} \end{cases}$$

For example, for the Manhattan distance between two tries, the traversal operator is defined as $\bigoplus \equiv \sum$ and the match function as

$$m(x,y) = \begin{cases} |x_c - y_c| & \text{if } x = y \\ |x_c| + |y_c| & \text{otherwise} \end{cases}$$

Figure 2(b) shows a snapshot of a traversal calculating the Manhattan distance. The match function at the nodes corresponding to the words {"barn"} and {"band", "bank"} is calculated as $|4| + |8|$, since a mismatch between n-grams implies addition of their counts, according to the definition of the Manhattan distance. By adapting the match function, one can calculate various similarity measures. We have implemented and applied the Canberra distance (29) and 'binarized" Manhattan distance, which have been used in previous work on 1-grams, the Czekanowski coefficient (30) and (second) Kulczynski coefficient (31), which are common non-metric similarity measures particularly suitable for description of sparse data. A brief description of these measures is given in Appendix A.1.

2.3 Unsupervised Anomaly Detection

Unsupervised anomaly detection is particularly suitable to the practical needs of intrusion detection, as it spares an administrator from the task of collecting data representative of normal activity. An unsupervised anomaly detection algorithm can be directly applied to a stream of data and is supposed to effectively discriminate between normal and anomalous patterns "on-the-fly". Furthermore, no extensive training using manually labeled data is required.

Because of its favorable properties, unsupervised anomaly detection has gained significant interest in recent work on intrusion detection (e.g. 32; 14; 33). The algorithms for unsupervised anomaly detection exploit differences in geometric features of anomalies and normal data. The algorithms explore local properties of the provided data as in the case of single-linkage clustering (32) and our k-nearest neighbor method *Zeta*, or analyze global properties as the simplified Mahalanobis distance (10) and the quarter-sphere SVM (34). A brief summary of these four algorithms used in our work is presented in Appendix A.2.

3 Experimental Results

In order to evaluate the proposed n-gram trie representation of network connections with respect to detection of unknown attacks and to gain insights into the nature of recovered syntactic and semantic information, we conducted experiments on two network traffic datasets. Specifically we are interested to clarify the following open questions:

1. How does the length of n-grams affect detection performance with respect to network protocols and attack types?
2. At what false-positive rate do we detect all instances of attacks present in the data?

We limit our experiments to the popular and text-based application-layer protocols HTTP, FTP and SMTP, which constitute a steady target of network attacks in the last decade.

Table 1. Remote-to-local attacks from DARPA 1999 dataset

HTTP attacks	FTP attacks	SMTP attacks
HTTP tunnel	`.rhost` upload	Sendmail exploit
PHF CGI attack	NcFTP exploit	Mail: Spoofed frame
	Password guessing	Mail: PowerPoint macro
		Mail: SSH trojan horse

3.1 Datasets

DARPA 1999 Dataset. This well-known dataset from an IDS evaluation conducted by the DARPA in 1999 (35) has been used in numerous publications and can be considered a standard benchmark for evaluation of IDS. Even though the DARPA 1999 dataset is known to suffer from several flaws and artifacts (12; 36; 37), especially the selection of attacks can be considered antiquated in comparison to modern security threats, it remains the only major dataset on which results can be reproduced.

As a preprocessing step, we randomly extracted 1000 TCP connections for each protocol from the first and third weeks of the data corpus representing normal data. We then selected all remote-to-local attacks present in the fourth and fifth weeks of the dataset. Table 1 lists these remote-to-local attacks.

PESIM 2005 Dataset. In order to overcome the problems of the DARPA 1999 dataset, we generated a second evaluation dataset named *PESIM 2005*. We deployed a combination of 5 servers using a virtual machine environment. The systems ran two Windows, two Linux and one Solaris operating systems and offered HTTP, FTP and SMTP services.

Normal network traffic for these systems was generated by members of our laboratory. To achieve realistic traffic characteristics we transparently mirrored news sites on the HTTP servers and offered file sharing facility on the FTP servers. SMTP traffic was artificially injected containing 70% mails from

Table 2. Remote-to-local attacks from PESIM 2005 data set

HTTP attacks	FTP attacks	SMTP attacks
HTTP tunnel	3COM 3C exploit	CMAIL Server 2.3 exploit
IIS 4.0 htr exploit	GlobalScape 3.x exploit	dSMTP 3.1b exploit
IIS 5.0 printer exploit	Nessus FTP scan	MS Exchange 2000 exploit
IIS unicode attack	ProFTPd 1.2.7. exploit	MailCarrier 2.51 exploit
IIS 5.0 webdav exploit	Serv-U FTP exploit	Mail-Max SMTP exploit
IIS w3who exploit	SlimFTPd 3.16 exploit	Nessus SMTP scan
Nessus HTTP scan	WarFTPd 1.65 pass exploit	NetcPlus SmartServer3 exploit
PHP script attack	WarFTPd 1.65 user exploit	Personal Mail 3.072 exploit
	WsFTPd 5.03 exploit	Sendmail 8.11.6 exploit
	WU-FTPd 2.6.1 exploit	

personal communication and mailing lists, and 30% spam mails received by 5 individuals. The normal data was preprocessed similarly to the DARPA 1999 dataset by random selection of 1000 TCP connections for each protocol from the data corpus. Attachments were removed from the SMTP traffic.

Attacks against the simulated services were generated by a penetration testing expert using modern penetration testing tools. Multiple instances of 27 different attacks were launched against the HTTP, FTP and SMTP services. The attacks are listed in Table 2. The majority of these attacks is part of the comprehensive collection of recent exploits in the Metasploit framework (38). Additional attacks were obtained from common security mailing lists and archives, such as Bugtraq and Packetstorm Security. The "PHP script attack" was introduced by the penetration testing expert and exploits insecure input processing in a PHP script.

3.2 Experimental Setup

The basic building block of our experiments are the incoming byte sequences of TCP connections. Each connection, normal or malicious, is transformed into a trie representing a respective language model. Our dataset thus consists of a set of tries computed over connection payloads.

Since our goal is the detection of unknown attacks, our algorithms are evaluated on randomly sampled mixtures of *unseen normal and attack data*. No explicit learning involving labeled attacks is performed.

On the other hand, the algorithms at our disposal require certain parameters to be set that affect their detection performance. Manual setting of such parameters usually results in tedious tuning of algorithms. Therefore, we precede the evaluation of algorithms with a validation stage, at which the best parameters are automatically selected based on an independent dataset. The crucial requirement in our setup is that *no data used at the validation stage is employed during evaluation*.

The evaluation criterion is the so-called *area under curve* (AUC) which integrates true-positive rates over a certain interval of false-positive rate, in our case $[0, 0.01]$. For the sake of statistical significance, the results are averaged over 30 validation/evaluation runs, comprising 1000 connections each.

3.3 Results

Best Measure/Detector Configuration. As it was previously mentioned, similarity measures induce various geometric properties which, in turn, are explored in different ways by anomaly detection methods. Hence, as a first step, we need to roughly establish what combinations of similarity measures and anomaly detectors perform best on n-gram tries for each network protocol in question. This can be done by averaging the AUC values for each measure/detector configuration over all values of n.

Table 3 lists the best three measure/detector configurations for the HTTP, FTP and SMTP protocols on both datasets. For all protocols similarity coefficients yield better accuracy than metric distances, which points to the sparse

Table 3. Best three measure/detector configurations for each protocol

Similarity measure	Anomaly detector	AUC
HTTP protocol		
Kulczynski coefficient	Quarter-sphere SVM	0.7807
Kulczynski coefficient	Zeta	0.7696
Czekanowski coefficient	Zeta	0.7580
FTP protocol		
Kulczynski coefficient	Zeta	0.7456
Kulczynski coefficient	Single-linkage clustering	0.5795
Czekanowski coefficient	Single-linkage clustering	0.5722
SMTP protocol		
Czekanowski coefficient	Single-linkage clustering	0.7561
Kulczynski coefficient	Zeta	0.7318
Kulczynski coefficient	Single-linkage clustering	0.7186

characteristics induced by high-order n-grams. For the HTTP protocol a global anomaly detector achieves the best performance, while for the other protocols local anomaly detectors perform best for varying length of n. In the remaining experiments we fix the measure/detector configuration to the best one for each network protocol.

Varying N-Gram Length. Previous results in natural language processing and host-based IDS indicate that the optimal n-gram length may vary for different applications (39; 19; 24). We now investigate if the same observation holds for n-gram models of TCP connection payloads.

We follow the same setup as in the selection of the optimal measure/detector configuration, except that results of individual values of n are reported using a fixed configuration. The results are shown in Fig. 3 for the DARPA 1999 dataset and Fig. 4 for the PESIM 2005 dataset, which display the ROC graphs for selected values of n.

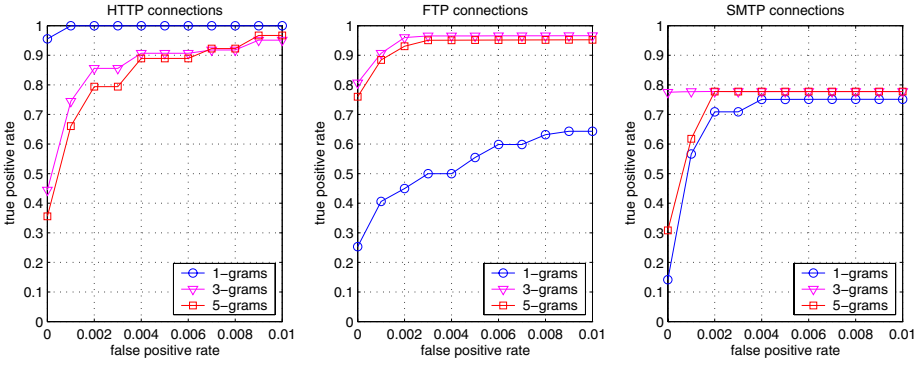


Fig. 3. ROC graphs for 1-, 3- and 5-grams (DARPA 1999)

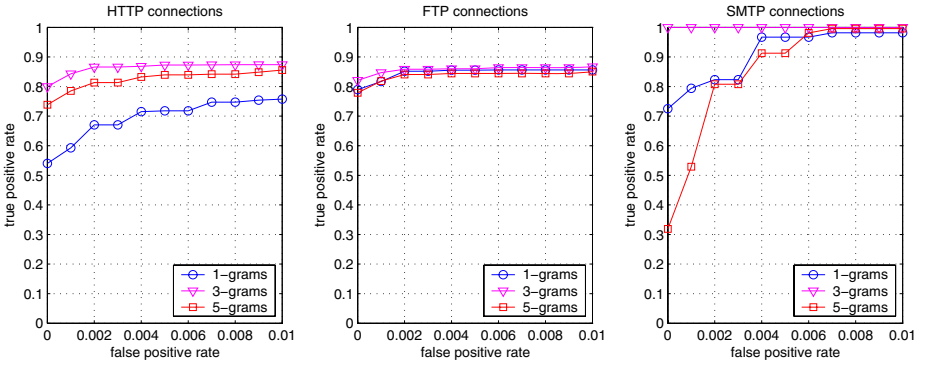


Fig. 4. ROC graphs for 1-, 3- and 5-grams (PESIM 2005)

Table 4. False-positive rates for detection of individual attacks (PESIM 2005)

Attack name	**# Instances**	n	**False-positive rate**
HTTP protocol			
HTTP tunnel	6	7	0.0231
IIS 4.0 htr exploit	3	1–7	0.0000
IIS 5.0 printer exploit	5	1–7	0.0000
IIS unicode attack	4	1	0.0987
IIS 5.0 webdav exploit	6	1	0.0322
IIS w3who exploit	3	2–7	0.0000
Nessus HTTP scan	6	7	0.0252
PHP script attack	5	2	0.0091
FTP protocol			
3COM 3C exploit	4	2–5	0.0000
GlobalScape 3.x exploit	4	1–2	0.0000
Nessus FTP scan	5	1–3	0.0000
ProFTPD 1.2.7 exploit	4	7	0.3448
Serv-U FTP exploit	4	2–3	0.0000
SlimFTPd exploit	4	2–5	0.0000
WarFTPd pass exploit	3	1–6	0.0000
WarFTPd user exploit	2	1–6	0.0000
WsFTPd exploit	4	2–5	0.0000
WU-FTPd exploit	4	6	0.0133
SMTP protocol			
CMAIL Server 2.3 exploit	4	3	0.0000
dSMTP 3.1b exploit	3	2	0.0003
MS Exchange 2000 exploit	2	3	0.0000
MailCarrier 2.51 exploit	4	3	0.0000
Mail-Max SMTP exploit	2	3	0.0000
Nessus SMTP scan	6	3-4	0.0000
NetcPlus SmartServer3 exploit	3	3	0.0000
Personal Mail 3.072 exploit	3	3	0.0000
Sendmail 8.11.6 exploit	4	3	0.0012

The detection performance varies significantly among the values of n for different protocols. In fact, it turns out that each of the three values considered in this experiment is optimal for some protocol. Apart from that, the overall accuracy of our approach is very encouraging, especially on the more recent PESIM 2005 dataset. For the best value of n, a detection rate above 80% was observed *with no false-positives* for the HTTP, FTP and SMTP protocols.

Analysis of Specific Attacks. One is always interested to know how well an IDS detects specific attacks in a dataset. As criterion for this experiment we considered the minimum false-positive rate at which all instances of an attack are detected. In addition, we record the optimal value of n for different attacks. The results are shown in Table 4.

One can clearly see that 18 from 27 attack types (66%) are perfectly recognized with no false positives. This demonstrates not only the high accuracy of n-gram-based anomaly detection but also its *wide coverage* within the attack spectrum.

Some interesting insights can be gained from the analysis of the optimal n for specific attacks. For several attacks, which are particularly easy to detect, the n-gram length is irrelevant. Noteworthy is the consistent optimality of $n = 3$ for several SMTP attacks which are also perfectly detected. For the attacks that are more difficult to detect, longer n-grams lengths seem to be prevalent. An extreme example is the ProFTPd exploit. This exploit uploads a malicious file to an FTP server. Since the file content is transfered over a data channel *not monitored by our system*, this attack can only be detected by chance in our setup.

4 From N-Grams to Words

The message from the experiments in the previous section may be somewhat confusing for a practitioner. One can see that longer n-grams bring improvement in detection performance in some cases, on the other hand, no consistency can be found across various attacks and protocols. How should one choose the right n beforehand if attacks are unknown?

The following extension of the n-gram model addresses this concern. Note that the semantics of natural languages is, in fact, defined in terms of words rather than n-grams. Words in a natural language are defined as consecutive character sequences separated by white-space symbols. Similarly, semantics of text-based protocols such as HTTP, FTP and SMTP can be captured by appropriately defined words and boundary symbols (16; 18). For our experiments we define the following global set of separator bytes that is used to tokenize payloads of HTTP, FTP and SMTP connections:

$$\{ \text{CR, LF, TAB, “ ”, “,”, “.”, “:” , “/” , “\&” } \}.$$

We are now about to discover another remarkable property of the trie representation of n-grams and the comparison method proposed in this paper: it can handle variable-length “grams” without any alteration!

We repeat the experiments under the same setup as the experiments on varying n-gram length using a stream of words instead of n-grams. The similarity

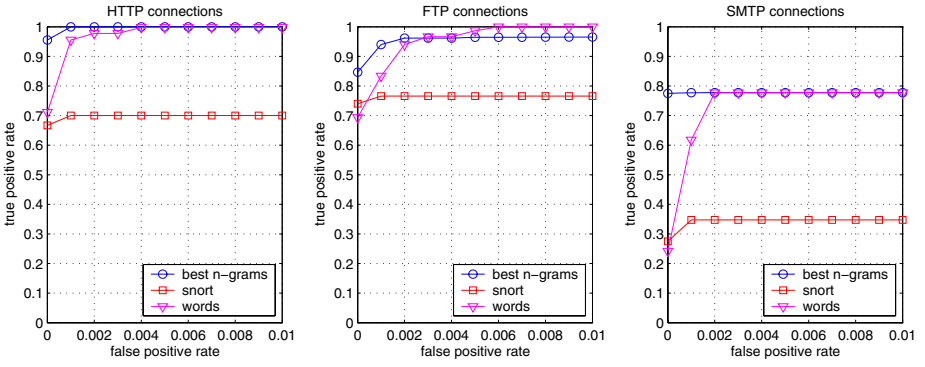


Fig. 5. ROC graphs for best n-grams and words (DARPA 1999)

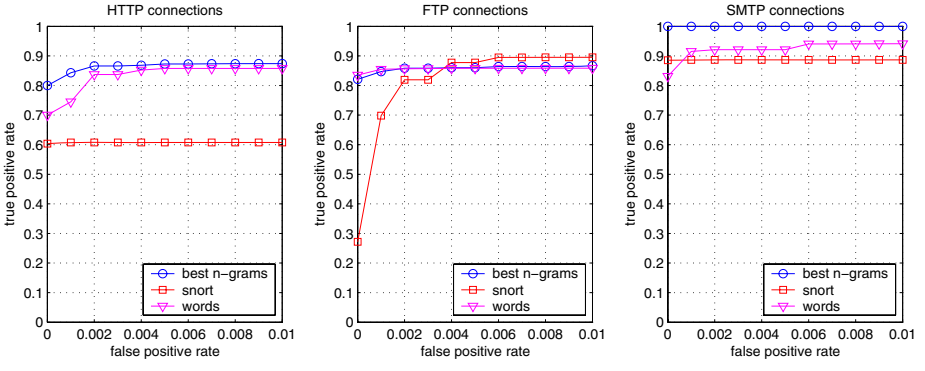


Fig. 6. ROC graphs for best n-grams and words (PESIM 2005)

measures applied in previous experiments are then computed over word frequencies, and the same optimal measure/detector configuration is used.

To emphasize the practical focus of this experiment, we compare the results of our models with the performance of the open-source signature-based IDS Snort (40) (Snort version 2.4.2, released on 28.09.2005 and configured with the default set of rules). The results are shown in Fig. 5 for the DARPA 1999 dataset and Fig. 6 for the PESIM 2005 dataset.

It can be seen that our word-based detector eventually yields the same accuracy as the best n-gram-based detector (at false-positive rates of about 0.5%). However, the initial ascent of the ROC curve is not as steep as for the best n-gram. This is the price one has to pay for being independent of the parameter n.

To our surprise, the n-gram and word models significantly outperformed Snort on the DARPA 1999 and PESIM 2005 dataset even though all included attacks except for the "PHP script" were known months before the release date of our Snort distribution. This result confirms a misgiving that signature-based IDS may fail to discover "fresh" attacks despite a major effort in the security community

to maintain up-to-date signature repositories. Noteworthy is the fact that Snort failed in our experiments due to two reasons. Some attacks were not detected because no appropriate signature was present, which is manifested by flat ROC graphs that never reach the 100% level. Other failures occurred due to minor variations in attack syntax. For example, one of the SMTP attacks was not discovered when an attacker replaced the initial "HELO" command with "EHLO", which is allowed by protocol specification and frequently used in practice.

5 Related Work and Discussion

Although advanced language models and tries have not been previously used in the context of network intrusion detection, they are well known in several other fields of computer science. Quite naturally, language models have been first developed by researchers in the fields of information retrieval and natural language processing – several decades before their relevance for intrusion detection was discovered. As early as mid-sixties, character n-grams were used for error correction in optical character recognition (41). Application of n-grams to text categorization was pioneered by Suen (42) and was followed by a large body of subsequent research (e.g. 25; 43; 44). Various similarity measures were used to compare n-gram frequencies, e.g. the inner product between frequency vectors (25) or Manhattan and Canberra distances (43). Recent approaches to text categorization advocate the use of kernel functions as similarity measures, which allows one to incorporate contextual information (45; 46; 39).

Re-discovery of n-gram models in the realm of host-based IDS began in the mid-nineties with the seemingly ad-hoc "sliding window" approach of Forrest et al. (19). Their main idea was to create a database of all possible n-grams in system call traces resulting from normal operation of a program. System call traces with a large degree of binary mismatch to the database were flagged as anomalous. In the ensuing work these ideas were extended through application of Hidden Markov Models (21), feed-forward and recursive neural networks (23), rule induction algorithms (47) and Support Vector Machines (14). As part of this evolution, trie and suffix tree data structure were introduced for storage and analysis of system call n-grams (24; 22; 48).

Application of n-gram models for network-based IDS originated in the idea of using byte histograms of packet payloads for statistical tests of anomality (11). A more advanced model was proposed by Wang and Stolfo, in which a simplified Mahalanobis distance is used over byte histograms to detect anomalous packet payloads (10; 9). To cope with varying packet length the byte histograms are conditioned on packet lengths and additional merging of adjacent models is used to control the size of the overall model.

The byte histograms of packet payloads by Wang and Stolfo can be seen as a particular case of an 1-gram model, whose similarity is measured using the simplified Mahalanobis distance. Compared to this approach, we incorporate advanced language models, such as high-order n-grams and words, and propose

an algorithm for linear-time computation of a wide range of similarity measures for such models using trie data structures.

Results of experiments conducted on the DARPA 1999 and PESIM 2005 datasets demonstrate the importance of higher-order n-grams for detection of recent network attacks. It is nonetheless difficult to determine an optimal length of n-gram models for particular attacks and protocols. This problem can be alleviated by considering language models based on words, using separators appropriate for protocol syntax. The accuracy of unsupervised anomaly detectors based on word models, as investigated in our experiments, is comparable to the accuracy of the best n-gram models. Furthermore, the system based on our language model significantly outperformed a recent version of the open-source IDS Snort equipped with the full standard set of signatures in a "plug-and-play" setup.

Acknowledgments

The authors gratefully acknowledge the funding from *Bundesministerium für Bildung und Forschung* under the project MIND (FKZ 01-SC40A) and would like to thank Stefan Harmeling, Sören Sonnenburg and Timon Schröter for fruitful discussions and support.

Bibliography

[1] Shannon, C., Moore, D.: The spread of the Witty worm. Proc. IEEE Symposium on Security and Privacy **2**(4) (2004) 46–50
[2] CERT: Advisory CA-2001-21: Buffer overflow in telnetd. CERT Coordination Center (2001)
[3] Rubin, S., Jha, S., Miller, B.: Language-based generation and evaluation of NIDS signatures. In: Proc. IEEE Symposium on Security and Privacy. (2005) 3–17
[4] Liang, Z., Sekar, R.: Automatic generation of buffer overflow attack signatures: An approach based on program behavior models. In: Proc. ACSAC. (2005) To appear.
[5] Kruegel, C., Kirda, E., Mutz, D., Robertson, W., Vigna, G.: Polymorphic worm detection using structural information of executables. In: Proc. RAID. (2005)
[6] Meier, M.: A model for the semantics of attack signatures in misuse detection systems. In: Proc. ISC. (2004) 158–169
[7] Eckmann, S., Vigna, G., Kemmerer, R.: STATL: An attack language for state-based intrusion detection. Journal of Computer Security **10**(1/2) (2002) 71–104
[8] Paxson, V.: Bro: a system for detecting network intruders in real-time. In: Proc. USENIX. (1998) 31–51
[9] Wang, K., Cretu, G., Stolfo, S.: Anomalous payload-based worm detection and signature generation. In: Proc. RAID. (2005)
[10] Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In: Proc. RAID. (2004) 203–222
[11] Kruegel, C., Toth, T., Kirda, E.: Service specific anomaly detection for network intrusion detection. In: Proc. Symposium on Applied Computing. (2002) 201–208
[12] Mahoney, M., Chan, P.: An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In: Proc. RAID. (2004) 220–237

[13] Mahoney, M., Chan, P.: PHAD: Packet header anomaly detection for identifying hostile network traffic. Technical Report CS-2001-2, Florida Institute of Technology (2001)
[14] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. In: Applications of Data Mining in Computer Security. Kluwer (2002)
[15] Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security **3** (2001) 227–261
[16] Mahoney, M., Chan, P.: Learning models of network traffic for detecting novel attacks. Technical Report CS-2002-8, Florida Institute of Technology (2002)
[17] Mahoney, M.: Network traffic anomaly detection based on packet bytes. In: Proc. ACM Symposium on Applied Computing. (2003) 346 – 350
[18] Vargiya, R., Chan, P.: Boundary detection in tokenizing netwok application payload for anomaly detection. In: Proc. ICDM Workshop on Data Mining for Computer Security. (2003) 50–59
[19] Forrest, S., Hofmeyr, S., Somayaji, A., Longstaff, T.: A sense of self for unix processes. In: Proc. IEEE Symposium on Security and Privacy, Oakland, CA, USA (1996) 120–128
[20] Hofmeyr, S., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security **6**(3) (1998) 151–180
[21] Warrender, C., Forrest, S., Perlmutter, B.: Detecting intrusions using system calls: alternative data models. In: Proc. IEEE Symposium on Security and Privacy. (1999) 133–145
[22] Marceau, C.: Characterizing the behavior of a program using multiple-length n-grams. In: Proc. NSPW. (2000) 101–110
[23] Ghosh, A., Schwartzbard, A., Schatz, M.: Learning program behavior profiles for intrusion detection. In: Proc. USENIX, Santa Clara, CA, USA (1999) 51–62
[24] Eskin, E., Lee, W., Stolfo, S.: Modeling system calls for intrusion detection with dynamic window sizes. In: Proc. DISCEX. (2001)
[25] Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199) (1995) 843–848
[26] de la Briandais, R.: File searching using variable length keys. In: Proc. AFIPS Western Joint Computer Conference. (1959) 295–298
[27] Fredkin, E.: Trie memory. Communications of ACM **3**(9) (1960) 490–499
[28] Knuth, D.: The art of computer programming. Volume 3. Addison-Wesley (1973)
[29] Emran, S., Ye, N.: Robustness of canberra metric in computer intrusion detection. In: Proc. IEEE Workshop on Information Assurance and Security, West Point, NY, USA (2001)
[30] Dice, L.: Measure of the amount of ecologic association between species. Ecology **26**(3) (1945) 297–302
[31] Sokal, R., Sneath, P.: Principles of numerical taxonomy. Freeman, San Francisco, CA, USA (1963)
[32] Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proc. ACM CSS Workshop on Data Mining Applied to Security. (2001)
[33] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection,. In: Proc. SIAM. (2003)
[34] Laskov, P., Schäfer, C., Kotenko, I.: Intrusion detection in unlabeled data with quarter-sphere support vector machines. In: Proc. DIMVA. (2004) 71–82

[35] Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks **34**(4) (2000) 579–595
[36] McHugh, J.: The 1998 Lincoln Laboratory IDS evaluation. In: Proc. RAID. (2000) 145–161
[37] McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. on Information Systems Security **3**(4) (2000) 262–294
[38] Moore, H.D.: The metasploit project – open-source platform for developing, testing, and using exploit code. http://www.metasploit.com (2005)
[39] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research **2** (2002) 419–444
[40] Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proc. LISA. (1999) 229–238
[41] Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Analysis and Machine Intelligence **22**(1) (2000) 36–62
[42] Suen, C.Y.: N-gram statistics for natural language understanding and text processing. IEEE Trans. Pattern Analysis and Machine Intelligence **1**(2) (1979) 164–172
[43] Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proc. SDAIR, Las Vegas, NV, USA. (1994) 161–175
[44] Robertson, A.M., Willett, P.: Applications of n-grams in textual information systems. Journal of Documentation **58**(1) (1998) 48–69
[45] Watkins, C.: Dynamic alignment kernels. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers, Cambridge, MA, MIT Press (2000) 39–50
[46] Leslie, C., Eskin, E., Noble, W.: The spectrum kernel: A string kernel for SVM protein classification. In: Proc. Pacific Symp. Biocomputing. (2002) 564–575
[47] Lee, W., Stolfo, S., Chan, P.: Learning patterns from unix process execution traces for intrusion detection. In: Proc. AAAI workshop on Fraud Detection and Risk Management, Providence, RI, USA (1997) 50–56
[48] Michael, C.: Finding the vocabulary of program behavior data for anomaly detection. In: Proc. DISCEX. (2003) 152–163
[49] Hamming, R.W.: Error-detecting and error-correcting codes. Bell System Technical Journal **29**(2) (1950) 147–160
[50] Anderberg, M.: Cluster Analysis for Applications. Academic Press, Inc., New York, NY, USA (1973)
[51] Harmeling, S., Dornhege, G., Tax, D., Meinecke, F., Müller, K.R.: From outliers to prototypes: ordering data. Neurocomputing (2006) in press.

A Appendix

A.1 Similarity Measures

A (dis)similarity measure is a binary function that maps x and y with component values x_i and y_i to a singular (dis)similarity score.

Metric Distances. The Canberra distance d_c is a normalized form of the Manhattan distance. It expresses metric characteristics and distance scores lie within

the range [0, 1]. The distance is suitable for histograms containing quantities and frequencies:

$$d_c(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i + y_i}$$

The "binarized" Manhattan distance d_b is similar to the Hamming distance (49). It is metric and maps the input vectors x and y to a binary space using the function b which returns 1 for non-zero values:

$$d_b(x, y) = \sum_{i=1}^{n} |b(x_i) - b(y_i)|$$

Similarity Coefficients. Similarity coefficients are often applied to binary data and express non-metric properties (50). These coefficients are constructed over four summation variables a, b, c and d. The variable a defines the number of positive matching components (1-1), b the number of left mismatches (0-1), c the number of right mismatches (1-0) and d the number of negative matches (0-0).

The coefficients can be extended to non-binary data by modification of these summation variables. The degree of matching between two components can be defined as $\min(x_i, y_i)$ and accordingly mismatches as differences from $\min(x_i, y_i)$:

$$a = \sum_{i=1}^{n} \min(x_i, y_i),\ b = \sum_{i=1}^{n} (x_i - \min(x_i, y_i)),\ c = \sum_{i=1}^{n} (y_i - \min(x_i, y_i))$$

The Czekanowski coefficient s_c measures the ratio between positive matching components and the sum of all components (30). In the extended form it can be expressed as following:

$$s_c(x, y) = \frac{2a}{2a + b + c} = \frac{2 \sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n} x_i + y_i}$$

The second Kulczynski coefficient s_k measures the ratio between positive matching components against the left- and right-hand side of mismatches (31). In the extended form the second Kulczynski coefficient is defined as following:

$$s_k(x, y) = \frac{1}{2}\left(\frac{a}{a + b} + \frac{a}{a + c}\right) = \frac{1}{2}\left(\frac{\sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n} x_i} + \frac{\sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n} y_i}\right)$$

A.2 Anomaly Detectors

Global Anomaly Detectors. The *simplified Mahalanobis distance* (10) determines the center of mass of data μ and the variance of each dimension σ_i in input space. The anomaly score is defined as the variance-scaled distance from x to μ:

$$m_{\mu,\sigma}(x) = \sum_{i=1}^{n} \frac{|x_i - \mu_i|}{\sigma_i}$$

The *quarter-sphere SVM* (34) is a kernel-based learning method that determines the center of mass of input data μ_ϕ in a high-dimensional feature space using a non-linear mapping function ϕ. The anomaly score is defined as the distance from $\phi(x)$ to μ_ϕ in feature space:

$$q_{\phi,\mu}(x) = ||\phi(x) - \mu_\phi||$$

Local Anomaly Detectors. Simplified *single-linkage clustering* (32) is a common clustering algorithm. Given a cluster assignment, the anomaly score is defined anti-proportional to the size of the cluster x is assigned to:

$$s_c(x) = \frac{1}{|c|} \text{ for } x \in c$$

Our new method *Zeta* is an anomaly score based on the concept of k-nearest neighbors and extends the outlier detection methods proposed in (51). The score is calculated as the mean distance of x to its k-nearest neighbors normalized by the mean inner-clique distance:

$$\zeta_k(x) = \frac{1}{k} \sum_{i=1}^{k} d(x, \text{nn}_i(x)) - \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} d(\text{nn}_i(x), \text{nn}_j(x))$$