J Comput Virol (2007) 2:243–256
DOI 10.1007/s11416-006-0030-0

ORIGINAL PAPER

Language models for detection of unknown attacks in network traffic

Konrad Rieck · Pavel Laskov

Received: 24 July 2006 / Accepted: 1 November 2006 / Published online: 19 December 2006

Abstract In this paper, we propose a method for network intrusion detection based on language models. Our method proceeds by extracting language features such as *n*-grams and words from connection payloads and applying unsupervised anomaly detection—without prior learning phase or presence of labeled data. The essential part of this procedure is linear-time computation of similarity measures between language models of connection payloads. Particular patterns in these models decisive for differentiation of attacks and normal data can be traced back to attack semantics and utilized for automatic generation of attack signatures.

Results of experiments conducted on two datasets of network traffic demonstrate the importance of high-order *n*-grams and variable-length language models for detection of unknown network attacks. An implementation of our system achieved detection accuracy of over 80% with no false positives on instances of recent remote-to-local attacks in HTTP, FTP and SMTP traffic.

1 Introduction

Detection of unknown attacks in network traffic is a long-standing issue on the wish-list of security practitioners. There exist numerous examples of previously unknown attacks, notably Internet worms [e.g. 1–3] and zero-day exploits [e.g. 4,5], that defeated common signature-based defenses, even though current applications and infrastructures for tracking vulnerabilities and their exploits claim to provide adequate protection by means of attack signatures. Furthermore, it often does not suffice for a signature to be available—deployed signatures must be managed, distributed and kept up-to-date by security administrators.

A large amount of previous work focused on anomaly detection in network traffic [e.g. 6–13]. The main hurdles on the way to its acceptance in practice are high false-positive rates and a lack of explainability and transparency in the detection process. The majority of previous approaches do not deliver sufficient accuracy in an acceptable range of false-positive rates and, furthermore, do not provide diagnostic information to help forensic analysis.

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 [6,14]. This information has proven to be useful for detection of malicious activity such as probes and port scans, yet it does not suffice to detect more dangerous attacks that exploit vulnerabilities of application-layer protocols and their implementations.

Recently, techniques of anomaly-based network intrusion detection have been proposed that analyze payloads of packets and connections [8,11–13,15–18]. 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.

K. Rieck (✉) · P. Laskov
Intelligent Data Analysis, Fraunhofer FIRST.IDA,
Kekuléstr. 7, 12489 Berlin, Germany
e-mail: konrad.rieck@first.fraunhofer.de

P. Laskov
e-mail: pavel.laskov@first.fraunhofer.de

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 the 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. Similar to natural language, most network protocols are manifested in a variety of "dialects" induced by implementation-specific interpretations and extensions of the protocol specification. Thus, protocols and natural languages possess grammatical structure and yet recovery of this structure is stymied by uncertainty and ambiguity.

In view of this linguistic analogy, one can see that detection of misuse and anomalous patterns amounts to learning syntactic and semantic fragments of an underlying protocol language. In contrast to individual characters, byte sequences reflect specific patterns of normal and attack data and can be used to interpret and evaluate alarms obtained using anomaly detection. Hence it is promising to apply the machinery of natural language processing to network intrusion detection.

Byte sequences can be represented by fixed-length and variable-length language models such as *n-grams* (sequences of n consecutive symbols) and *words* (sequences tokenized using a set of delimiter symbols). N-grams have been extensively used in host-based intrusion detection for modeling traces of system calls [e.g. 19–24], but until recently have not been applied in the context of network intrusion detection for $n > 1$. Tokenized words have been used for anomaly detection using rule-based learning [e.g. 10,11]. The recently proposed method [25] builds on the ideas from host-based intrusion detection and uses a Bloom filter to represent high-order n-grams of normal and malicious packet payloads. The approach pursued in this paper differs from previous work in that we use a geometric representation of high-order n-grams, which allows us to perform anomaly detection without building a global profile for normal events. The main technical difficulty that needs to be addressed for geometric analysis of byte sequences is:

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

Language models extracted from connection payloads enable the computation of pairwise similarity between connections—an essential procedure for application of *unsupervised anomaly detection algorithms*. Hence, we focus our attention on methods for efficient computation of similarity measures between n-grams and words. To address this problem we propose (a) a representation of n-grams and words using *tries* and (b) a linear-time method for comparison of tries.

The rest of this paper is organized as follows: in Sect. 2 language models, similarity measures and efficient methods for their computation are introduced. Section 3 covers unsupervised anomaly detection algorithms suitable for language models. Experiments on two datasets of network traffic and an evaluation of different language models are given in Sect. 4. Section 5 illustrates interpretation of network anomalies using language models and automatic generation of attack signatures. In Sect. 6 we review related work and conclude this contribution.

2 Language models

In order to encapsulate and access semantic and syntactic constructs of network connections using language models, one needs to cast connection payloads to a formal representation. Similarly to sentences from natural languages, connection payloads can be characterized by simple language models such as n-grams and words, which have proven to unveil discriminative information for text categorization and classification in natural language processing [26–29].

2.1 N-gram and word models

An incoming connection payload x corresponds to consecutive sequence of symbols from an alphabet Σ. The content of x can be modeled as a set of possibly overlapping subsequences w taken from a language $L \subseteq \Sigma^*$. The length of w is denoted by n.

- The model of *n-grams* can be derived by defining $L = \Sigma^n$, the language containing all sequences of fixed length n.
- Provided a set of delimiter symbols $D \subset \Sigma$, the model of *words* is defined as $L = (\Sigma \setminus D)^*$ where every $w \in L$ subsequence of x is delimited by symbols from D.

The chosen language L constitutes the basis for calculating similarity between network connections. Given a connection payload x and a language L, a geometric embedding into a feature space is performed by

Table 1 Kernel and distance functions for language models

Kernel and distance functions	
Linear kernel	$\sum_{w\in L} \phi_w(x)\phi_w(y)$
Canberra distance	$\sum_{w\in L} \frac{\lvert\phi_w(x)-\phi_w(y)\rvert}{\phi_w(x)+\phi_w(y)}$
Geodesic distance	$\arccos\left(\sum_{w\in L} \phi_w(x)\phi_w(y)\right)$
Jensen distance	$\sum_{w\in L} (H(\phi_w(x),\phi_w(y)) + H(\phi_w(y),\phi_w(x)))$

Table 2 Similarity coefficients for language models

Similarity coefficients	
Jaccard	$\frac{a}{a+b+c}$
Czekanowski	$\frac{2a}{2a+b+c}$
Sokal–Sneath	$\frac{a}{a+2(b+c)}$
Kulszynski	$\frac{1}{2}\left(\frac{a}{a+b}+\frac{a}{a+c}\right)$

calculating $\phi_w(x)$ for every $w \in L$ appearing in x. Usually the function $\phi_w(x)$ returns the frequency of w in x, however, other definitions returning a count or a binary flag for w are possible. The vector of all $\phi_w(x)$ embeds the connection payload x in a high-dimensional feature space, whose dimensions correspond to all $w \in L$. A detailed discussion of language models and their geometric embedding for natural language processing is available in [27,30,31].

2.2 Similarity measures for language models

By utilizing the geometric representation induced through ϕ, one can adapt classical, vectorial similarity measures, such as kernel and distance functions, to operate on language models of connection payloads. Table 1 lists common distance and kernel functions some of which have been applied in the domain of network intrusion detection [9,32,33]. The Jensen distance in Table 1 is defined using an entropy-like function $H(a,b) = a\log(2a/(a+b))$.

Another way for measuring similarity in vector spaces are so called similarity coefficients [e.g. 34,35]. These coefficients are non-metric and have been primarily used on sparse binary data, which makes them suitable for high-order n-gram and word models. Similarity coefficients are constructed using three summation variables a, b and c. The variable a contains the number of positive matches (1–1), b the number of left mismatches (0–1) and c the number of right mismatches (1–0). The most common similarity coefficients are given in Table 2.

Similarity coefficients can be extended to non-binary data by modification of the summation variables. The degree of match for a sequence $w \in L$ can be defined as $\min(\phi_w(x),\phi_w(y))$ and the respective mismatches are defined as deviations thereof

$$a = \sum_{w\in L} \min(\phi_w(x),\phi_w(y))$$

$$b = \sum_{w\in L} \left[\phi_w(x) - \min(\phi_w(x),\phi_w(y))\right]$$

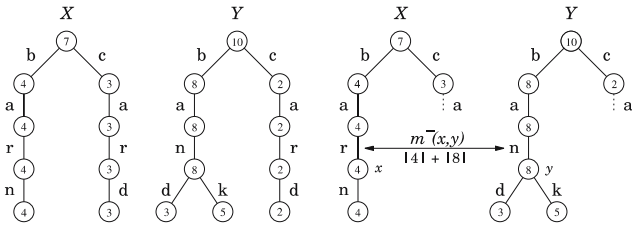
$$c = \sum_{w\in L} \left[\phi_w(y) - \min(\phi_w(x),\phi_w(y))\right]$$

Due to the high dimensionality of the induced vector space special algorithms are required for efficient computation of the proposed similarity measures. A linear-time algorithm based on trie data structures is introduced in the following section.

2.3 Efficient computation of similarity measures

The classical scheme for comparing language models utilizes indexed tables or in the more general case hash tables [e.g. 27]. Subsequences w of length n extracted from a connection payload x and the corresponding values $\phi_w(x)$ are stored in the bins of a table. Assuming the size of the table is fixed at M, it takes on average $\Theta(nM)$ to compare two tables: one needs to loop over all M bins, checking for matching and mismatching sequences. For small n indexed tables are efficient for comparison of language models, e.g. as used for $n = 1$ in [8,13,18]. For larger n an indexed representation becomes infeasible and hash tables or Bloom filters need to be utilized due to the exponentially growing number of possible sequences. However, to avoid hash collisions, a high value of M must be chosen in advance, which constitutes the main computational drawback of any hash table approach.

A better alternative for comparing language models are trie data structures [36–38]. A trie is an N-ary tree, whose nodes are N-place vectors with components corresponding to the characters of an alphabet Σ with $N = |\Sigma|$. Fig. 1a shows two tries X and Y containing the 4-grams {"barn", "card"} and {"bank", "band", "card"}. The nodes of a trie are augmented to carry attributes reflecting $\phi_w(x)$ for each sequence w extracted from the connection payload x. For example the left trie X in Fig. 1a holds $\phi_{\text{"barn"}}(x) = 4$ and $\phi_{\text{"card"}}(x) = 3$.



(a) Tries for language models **(b)** Comparison of tries

Fig. 1 Trie data structures **a** and their comparison **b**

Comparison of two tries can be carried out by enumerating matching and mismatching sequences. 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. Since only a linear number of n-grams or words can be extracted from a connection payload x, the worst-case run time is $O(n|x|)$. An advantage of the trie data structure comes into play if the provided alphabet is large and a lot of mismatches occur. The traversal discovers mismatching words after passing the first few symbols and omits further unnecessary comparisons.

Computation of similarity measures using tries is straight forward: during parallel traversal the values of $\phi_w(x)$ and $\phi_w(y)$ stored in the nodes are aggregated according to the definition of a chosen similarity measure. Figure 1b shows a snapshot of a traversal calculating the Manhattan distance. A mismatch m^- at the nodes corresponding to the words {"barn"} and {"band", "bank"} corresponds to $|\phi_{\text{"barn"}}(x) - 0| + |0 - \phi_{\text{"ban"}}(y)|$ and results in the calculation $|4| + |8|$. A detailed discussion on computation of various similarity measures for sequential data is given in [39,40].

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* learning algorithm can be directly applied to a stream of data and is supposed to effectively discriminate between normal and anomalous patterns "on-the-fly" without extensive training or manually labeled data. Because of these favorable properties, unsupervised anomaly detection has gained significant interest in recent work on intrusion detection [e.g. 9,32,41–43].

Algorithms for unsupervised anomaly detection exploit differences in geometric representations of anomalies and normal data. They differ in the concrete notion of normality and abnormality.[1] Some explore local properties of the provided data for determining outliers, e.g. single-linkage clustering [32] and our k-nearest neighbor method Zeta [44], others analyze global properties, e.g. the simplified Mahalanobis distance [13] and quarter-sphere SVM [42], to identify instances deviating from the mass of data.

The language models and similarity measures introduced in Sect. 2 enable one to define geometric distances between connection payloads, which indirectly reflect semantic differences between attacks and normal data. Since many of the unsupervised anomaly detection algorithms are defined in terms of distances, one can thus apply them for network intrusion detection. Following is a brief description of four algorithms applied on connection payloads in this paper.

The *simplified Mahalanobis distance* [13] is a global anomaly detection method that 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 d from point x to μ

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

The *quarter-sphere SVM* [42] is a kernel-based learning method that determines the center of mass of input data μ_φ in a feature space using a non-linear mapping function φ. Herein, the non-linear function φ does not necessary correspond to the embedding function $\phi_w(x)$, e.g. in case of the RBF kernel function. The anomaly score is defined as the distance d from $\varphi(x)$ to μ_φ in the non-linear feature space

$$q_{\varphi,\mu}(x) = d(\varphi(x), \mu_\varphi)$$

Simplified *single-linkage clustering* [32] is a common distance-based clustering algorithm. Given a cluster assignment, the anomaly score is defined to be inversely proportional to the size of the cluster C the point x is assigned to, so that small clusters yield high anomaly scores

$$s_C(x) = \frac{1}{|C|} \quad \text{for } x \in C$$

Our method Zeta [44] is an anomaly score based on the concept of k-nearest neighbors; it extends the outlier detection methods proposed in [45,46]. The score is calculated as the mean distance of x to its k-nearest

[1] We denote the property of deviating from normal as *abnormality* and refer to a data instance deviating from normal as *anomaly*.

neighbors normalized by the mean inner-clique distance.

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

The first term emphasizes the points that lie far away from its neighbors, whereas the second term discounts abnormality of points with wide neighborhood cliques.

4 Experimental results

In order to evaluate the proposed representation of network connection payloads using language models 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 datasets of network traffic. Specifically, we are interested to clarify the following open questions:

(1) How does the length of fixed-length models such as n-grams affect detection performance with respect to network protocols and attack types?
(2) At what false-positive rate does one detect all instances of an unknown attack present in network traffic?
(3) How does detection accuracy of fixed-length models compare to variable-length models such as words?

We limit our experiments to the popular text-based application-layer protocols HTTP, FTP and SMTP and remote attacks against the corresponding services.

4.1 Network traffic datasets

DARPA 1999 dataset. This well-known dataset from an IDS evaluation conducted by the DARPA in 1999 [47] 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 [10,48,49], 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 samples, each comprising 1,000 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 3 lists these remote-to-local attacks.

Table 3 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

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 five 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 personal communication and mailing lists, and 30% spam mails received by five individuals. The normal data was preprocessed similarly to the DARPA 1999 dataset by random selection of samples each comprising 1,000 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 4. The majority of these attacks is part of the comprehensive collection of recent exploits in the Metasploit framework [50]. 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.

4.2 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.

Table 4 Remote-to-local attacks from PESIM 2005 dataset

HTTP attacks	FTP attacks	SMTP attacks
HTTP tunnel	3COM 3C exploit	CMAIL Server 2.3 exp.
IIS 4.0 HTR exploit	GlobalScape 3.x exploit	dSMTP 3.1b exploit
IIS 5.0 printer exp.	Nessus FTP scan	MS Exchange 2000 exp.
IIS unicode attack	ProFTPd 1.2.7. exploit	MailCarrier 2.51 exploit
IIS 5.0 WebDAV exp.	Serv-U FTP exploit	Mail-Max SMTP exploit
IIS w3who exploit	SlimFTPd 3.16 exploit	Nessus SMTP scan
Nessus HTTP scan	WarFTPd 1.65 exp. 1	NetcPlus Server exploit
PHP script attack	WarFTPd 1.65 exp. 2	Personal Mail 3.x exploit
	WsFTPd 5.03 exploit	Sendmail 8.11.6 exploit
	WU-FTPd 2.6.1 exploit	

Since our goal is the detection of unknown attacks, our algorithms are evaluated on randomly sampled mixtures of *unseen normal and attack data* containing 2 to 14% malicious connections. 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 independent samples of our datasets. The crucial requirement in our setup is that *no data used at the validation stage is employed during evaluation*.

The evaluation criterion for all of the following experiments is the so-called *area under curve* ($\text{AUC}_{0.01}$) 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 for experiments are averaged over 30 validation/evaluation runs on randomly drawn samples comprising 1,000 connections each.

Experiment 1: Best measure/detector configuration

As it was previously mentioned, similarity measures induce various geometric properties which, in turn, are explored in different ways by unsupervised anomaly detection methods. Hence, as a first step, we need to roughly establish what combinations of similarity measures and anomaly detectors perform best on language models of connection payloads. We restrict this evaluation to the class of fixed-length models of n-grams and average the $\text{AUC}_{0.01}$ values for each measure/detector configuration over values of n from 1 to 7.

Table 5 lists the best measure/detector configurations averaged over all protocols on both datasets. One can see that the Zeta algorithm prevails among the best overall configurations, using various similarity measures. The best configuration is achieved with the Kulczynski coefficient calculated using a binary embedding function ϕ_w. The quarter-sphere SVM yields an overall second best configuration, yet it was bounded to a linear kernel. The same configurations scored among the six best (in a slightly different order) on the individual datasets, although the attacks are almost completely different.

In contrast to the other applied anomaly detection methods, the Zeta anomaly score builds on the concept of k-nearest neighbors, which is known for its ability to cope with sparse and heterogeneous data distributions occuring in the embedding space of high-order n-grams [45]. Furthermore, the Kulczynski coefficient has been specifically designed for comparison of sparse and binary data [35].

In the remaining experiments we fix the measure/detector configuration to the Kulczynski coefficient as similarity measure and the Zeta anomaly score as unsupervised anomaly detection algorithm.

Experiment 2: 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 and datasets [19,24,51,52]. We now investigate if the same observation holds for n-gram models of TCP connection payloads.

We follow the same setup as in the previous 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. 2 for the DARPA 1999 dataset and Fig. 3 for the PESIM 2005 dataset, which display the ROC graphs for selected values of n.

Table 5 Best overall measure/detector configuration

Similarity measure	ϕ_w type	Anomaly detector	$AUC_{0.01}$
Kulczynski coefficient	Binary	Zeta anomaly score	0.772
Linear kernel	Freq.	Quarter-sphere SVM	0.756
Kulczynski coefficient	Freq.	Zeta anomaly score	0.737
Czekanowski coefficient	Freq.	Zeta anomaly score	0.737
Jensen distance	Freq.	Zeta anomaly score	0.727
Geodesic distance	Freq.	Zeta anomaly score	0.712

The detection performance varies significantly among the values of n for different protocols. In fact, it turns out that each of the three values for n considered in this experiment is optimal for some protocol. 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.

Experiment 3: Analysis of specific attacks

In order to investigate why no optimal n could be established in the previous experiment we extend our analysis to the detection performance on individual network attacks. As criterion for this experiment we consider the minimum false-positive rate at which all instances of an unknown attack can be identified. In addition, we record the optimal value of n that yields the minimum false-positive rate. The results are shown in Table 6.

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 fixed-length models for 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. 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 transferred over a data channel *not monitored by our system*, this attack can only be detected by chance in our setup.

In practice, a security administrator will not have an opportunity to experiment with an optimal n-gram length in order to tune a system in various ways for different kinds of attacks. Furthermore, optimal values of n obtained using a dataset only reflect properties of specific data and might be insufficient for detection of novel attacks [52]. Therefore, techniques that avoid pre-setting of a fixed n-gram length should be investigated.

Table 6 False-positive rates for detection of individual attacks (PESIM 2005)

Attack name	#	n	False-positive rate
HTTP protocol			
HTTP tunnel	6	1	0.0000
IIS 4.0 HTR exploit	3	1–2, 7	0.0000
IIS 5.0 printer exploit	5	1–7	0.0000
IIS unicode attack	4	4	0.0016
IIS 5.0 WebDAV exploit	6	1–2	0.0000
IIS w3who exploit	3	3–5, 7	0.0000
Nessus HTTP scan	6	3	0.0571
PHP script attack	5	4	0.0184
FTP protocol			
3COM 3C exploit	4	2–5	0.0000
GlobalScape 3.x exploit	4	1	0.0000
Nessus FTP scan	5	1–3	0.0000
ProFTPD 1.2.7 exploit	4	7	0.6798
Serv-U FTP exploit	4	2–5	0.0000
SlimFTPd exploit	4	2–6	0.0000
WarFTPd pass exploit	3	1–6	0.0000
WarFTPd user exploit	2	1–5	0.0000
WsFTPd exploit	4	2–6	0.0000
WU-FTPd exploit	4	7	0.0273
SMTP protocol			
CMAIL Server 2.3 exploit	4	1–3, 5	0.0000
dSMTP 3.1b exploit	3	1	0.0002
MS Exchange 2000 exploit	2	2–6	0.0000
MailCarrier 2.51 exploit	4	1, 3	0.0000
Mail-Max SMTP exploit	2	1	0.0003
Nessus SMTP scan	6	1–6	0.0000
NetcPlus SmartServer3 exploit	3	1–3, 5	0.0000
Personal Mail 3.072 exploit	3	1–3, 5–6	0.0000
Sendmail 8.11.6 exploit	4	5	0.0040

Experiment 4: Combined and variable-length models

One possibility to avoid a pre-defined n-gram length is to run n anomaly detectors in parallel and combine their scores. A natural criteria for combination of anomaly scores of multiple identical detectors operating on similar features is the maximum value of the different

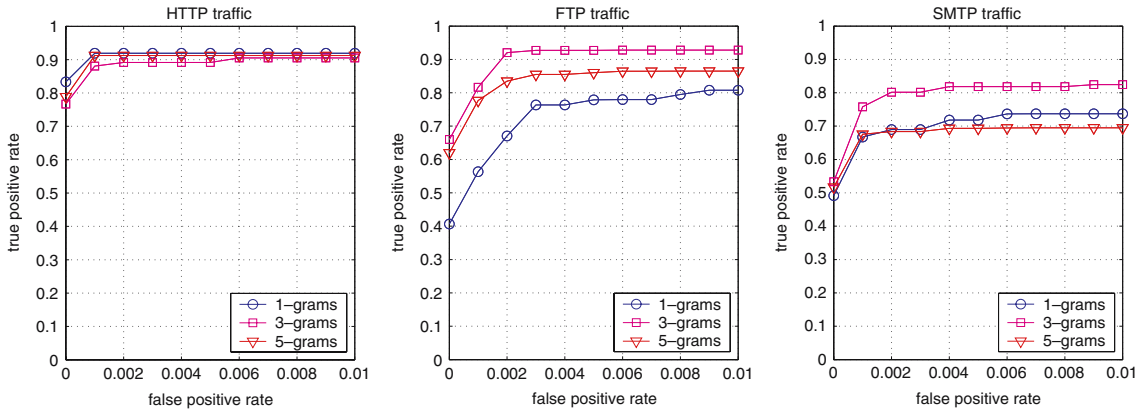


Fig. 2 ROC graphs for 1-, 3- and 5-grams (DARPA 1999)

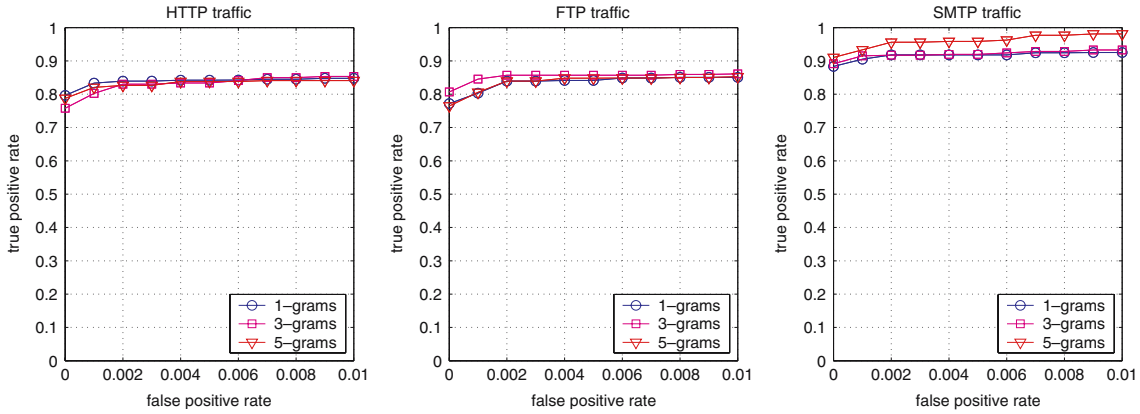


Fig. 3 ROC graphs for 1-, 3- and 5-grams (PESIM 2005)

anomaly scores. The main disadvantage of the combined scores approach is that one has to run as many detectors as the maximal n-gram length.

A less computationally intensive alternative is to use the variable-length model of words instead. The semantics of text-based protocols such as HTTP, FTP and SMTP is characterized by the presence of boundary symbols [11,15] which can be used as delimiters for definition of words. For our experiments we define the following global set of separator bytes that is used to tokenize connection payloads of HTTP, FTP and SMTP connections

```
CR LF TAB SPC , . : / & ? = ( ) [ ]
```

In order to avoid over-long sequences resulting from binary attack patterns such as shell-codes for buffer or heap overflows, we restrict the total length of words to 16 bytes and automatically split sequences.

We repeat the experiments under the same setup as the experiments on varying n-gram length using combined scores from multiple detectors and words extracted from connection payloads. 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 [53] (Snort version 2.4.2, released on 28.09.2005 and configured with the default set of rules). The results are shown in Fig. 4 for the DARPA 1999 dataset and Fig. 5 for the PESIM 2005 dataset.

It can be seen that although the word-based detector is somewhat less accurate than the detector combining multiple n-gram lengths, the marginal decrease in accuracy can be considered acceptable in comparison to n times smaller computational load.

To our surprise, both variable-length models significantly outperformed Snort on the DARPA 1999 and PESIM 2005 dataset even though all included attacks except for the "PHP script attack" were known months before the release date of the 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.

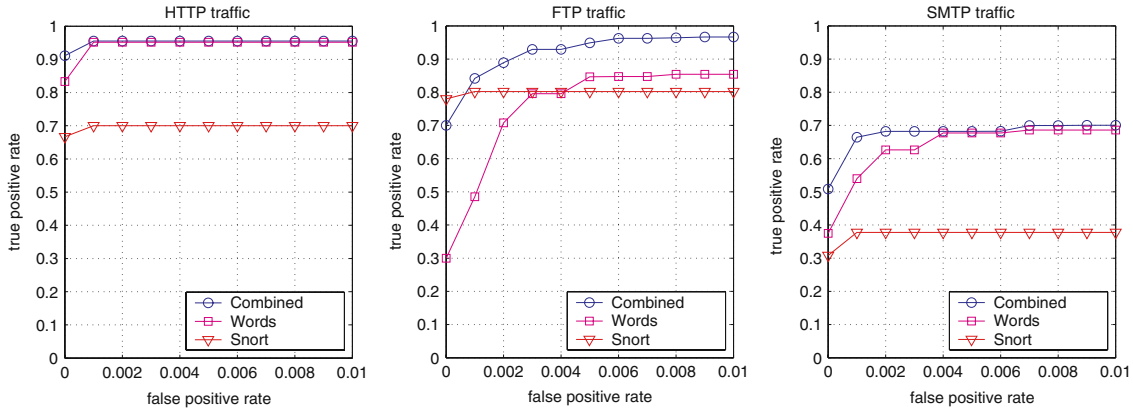


Fig. 4 ROC graphs for variable-length models versus Snort (DARPA 1999)

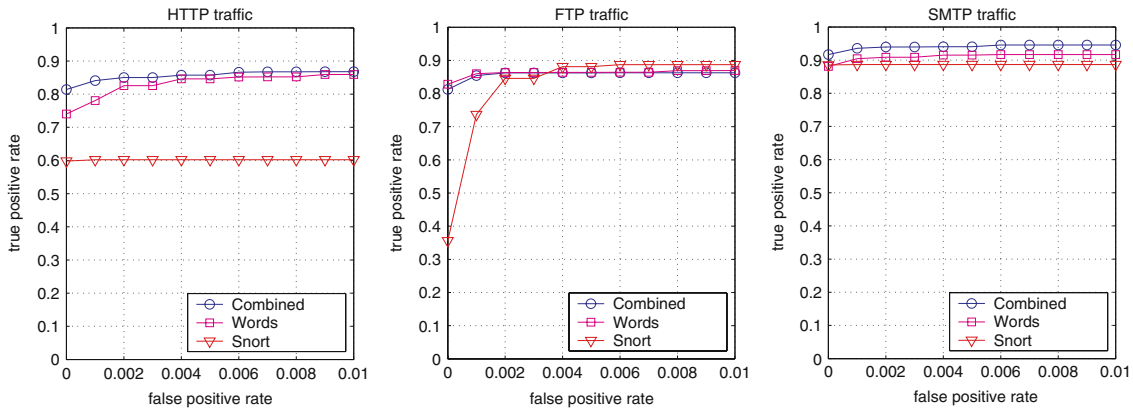


Fig. 5 ROC graphs for variable-length models versus Snort (PESIM 2005)

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 conforms to the protocol specification and is frequently used in practice.

5 Interpretation of language models

The four experiments from the previous section demonstrate the detection performance of unsupervised anomaly detection using language models of connection payloads. Especially the high detection accuracy achieved at low false-positive rates and the absence of a prior training phase makes the presented method a reasonable alternative to classical signature-based intrusion detection.

In practice, however, intrusion detection systems must not only flag malicious events, but also equip alarms with information necessary for categorization and assessment of security incidents. One reason for the lack of mature anomaly detection systems in the current security market is their inability to support interpretation and explanation of reported anomalies.

We address this problem using language models and illustrate that their advantages for detection of unknown attacks are indeed rooted in the ability of longer byte sequences to capture important attack semantics. We extend our intrusion detection method by (a) a diagnostic visualization of anomalous patterns and (b) a technique for automatic generation of signatures from detected anomalies.

5.1 Visualization of anomalous patterns

An anomalous connection payload identified using language models is represented by a set of extracted sequences, such as n-grams or words. By computing the differences in frequencies between the sequences in such

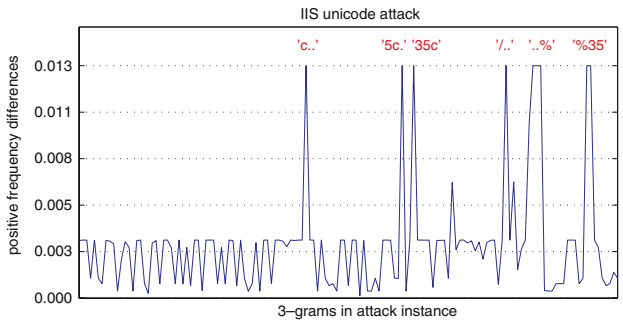


Fig. 6 Three-gram frequency differences for the IIS unicode attack

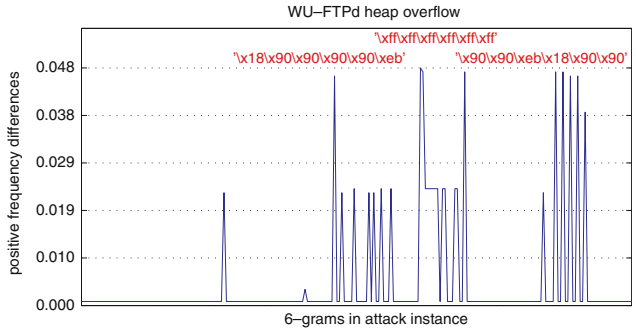


Fig. 7 Six-gram frequency differences for the WU-FTPd 2.6.1 exploit

a connection and normal network traffic, one obtains a *frequency difference plot*. The positive peaks in the truncated difference plot (the negative differences are of no particular value) are sequences or patterns that contribute the most to dissimilarity of the connection from normal traffic.

Figure 6 shows a 3-gram frequency difference plot for an instance of the IIS unicode attack. The attack exploits a vulnerability in the Microsoft IIS server, whose path-parsing logic fails to detect directory traversals hidden in multiple unicode encodings [54]. An example of the attack is given below:

```
GET /scripts/..%%35c../..%%35c../..%%35c..
    /..%%35c../..%%35c..
    /winnt/system/cmd.exe?/c+dir+c: HTTP/1.0
```

Strong positive peaks in the plot correspond to the 3-grams "35c", "/.." and "%35". These 3-grams manifest the 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).

A second example of this visualization is given in Fig. 7, which shows 6-gram frequency differences of an anomalous connection containing a WU-FTPd 2.6.1 exploit. The attack is manifested in two patterns: `0xffffffff` and `0x9090eb18`. The first pattern `0xffffffff` stems from an exploitation technique used to manipulate the control flow in the heap of the GNU C library [55]. The second pattern `0x9090eb18` is part of a so called "NOP sled" which corresponds to non-functional instructions at the prefix of a shell-code. The pattern contains, beside the common byte `0x90` (x86 assembler for no-operation), the instruction `0xeb18` (x86 assembler for jumping 24 bytes) which skips bytes corrupted by the heap memory management.

Both examples illustrate that frequency differences of language models between anomalous connections and normal network traffic constitute a diagnostic tool which emphasizes patterns decisive for reported anomalies and improves the assessment of security incidents by a security practitioner.

5.2 Language models for signature generation

A crucial step towards integration of anomaly detection methods into practice is interlinkage with existing signature-based systems. Once an unknown attack has been identified by anomaly detection, a further step is to generate a corresponding attack signature. Usually these signatures are manually crafted during a time-consuming inspection of multiple attack instances. Recently several approaches for automatic generation of signatures have been proposed to overcome this problem [e.g. 56–59]. In the following section, we demonstrate that language models can be applied in a similar manner for automatic generation of signatures.

In our approach an attack is represented by a trie containing sequences of a connection payload extracted with respect to a language model. By merging the tries of multiple instances of the same attack and pruning subtrees that reflect patterns occurring only in single connections, we construct a general model for a particular attack. We refer to this merged trie as an *A-signature*. Similarly, we can merge and prune tries containing sequences of normal connection payloads and construct a merged trie comprising characteristic patterns of normal network traffic. By assigning $+1$ to attack and -1 to normal sequences and adding the merged tries of attacks and normal connections, we can build *AN-signatures* that cover patterns occurring in either malicious or normal connection payloads—but not in both.

To evaluate the concept of A-signatures and AN-signatures, we conducted experiments on the PESIM 2005 dataset with the language model of words that proved effective for unsupervised anomaly detection in Sect. 4.2. We split the PESIM 2005 dataset into

```
Attack patterns (+1)
scripts exe cmd c+dir+c \ % %%35c

Normal patterns (−1)
utf-8 static rv p png package like keep-alive img image
i686 h http htm html gzip guides gif en en-us en-US d de
deflate ap application X X11 U User-Agent Safari Referer
PPC OS Mozilla Mac Macintosh Linux Keep-Alive KHTML
If-Modified-Sinc ISO-8859-1 Host Gecko GMT Firefox Debian
Cookie Connection Aug AppleWebKit Accept Accept-Language
Accept-Encoding Accept-Charset 8 7 6 5 4 42 41 412 40 4-2
3 30 300 2 20 2005 19 192 16 168 15 10 102 1020 07 00 *
```

Fig. 8 AN-signature for the IIS unicode attack

```
Attack patterns (+1)
X-LINK2STATE CHUNK=AAAAAAAAAA CCCCCCCCCCC CCCCCCCCCCCCC
BBBBBBBBBBBBBBB AAAAAAAAAAAAAA AAAAAAAAAAAAAAA

Normal patterns (−1)
z ze zebster w wo wonder wonderland with u us use user
userid t th te s so solaris solaris8 r p o n m ma l local
i in id h f fr from fo for e d dhcp c co ch b by a T To
TO S Subject R Re Received RC RCPT Q QUIT P Post Postfix
O Oct M Mo Mon Me Message MA MAIL F From FROM E EHLO D Da
Date DATA C:0.0, Co CEST 5 50 502 4 3 2 20 200 2005 1 11
10 0 - + +0 +0200
```

Fig. 9 AN-signature for the MS Exchange 2000 exploit

two distinct partitions and use the first one for construction of merged trie signatures. Figures 8 and 9 illustrate the automatically generated AN-signatures for the IIS unicode attack and the MS Exchange 2000 exploit.

The AN-signature in Fig. 8 constructed using words as language model contains exactly the patterns essential for the semantics of the IIS unicode attack [54], such as `cmd`, `c+dir+c` and `%%35c`. Words present in the exploit such as `GET` or `HTTP` have been automatically removed from the AN-signature as they also occur in almost any normal HTTP connection payload. Normal patterns of the signature reflect common keywords of the HTTP protocol such as `Accept` and `User-Agent`. Some of the normal words, however, result from the specific network environment used during generation of the dataset. A high percentage of Apple and Linux systems is manifested in the words `Safari`, `Macintosh`, `Debian` and `Linux`.

Figure 9 shows an AN-signature generated for the MS Exchange 2000 exploit [60]. Beside sequences for provoking a buffer overflow such as multiple `A`'s, `B`'s and `C`'s the attack patterns contain the keywords `X-LINK2STATE` and `CHUNK` which exactly correspond to the heap overflow vulnerability of the MS Exchange 2000 server. The normal patterns automatically extracted cover several common keywords of the SMTP protocol, but also reflect words specific to the network environment of the dataset, such as the hostnames `zebster`, `solaris8` and `wonderland`.

The detection performance of the automatically generated A-signatures and AN-signatures was evaluated on the second partition of the PESIM 2005 dataset. As detection score we defined the number of matching attack patterns minus the number of contained normal patterns in connection payloads. Figure 10 shows ROC graphs for the protocols HTTP, FTP and SMTP. The A-signatures detected only 30–60% of attacks since only some semantic patterns could be encapsulated in the corresponding language models, while the AN-signatures covered 80–100% of attacks with no false-positives. By modeling normal and attack characteristics in AN-signatures detection accuracy can be greatly improved in a specific network environment, however, in practice signature generation using language models should be applied in a *semi-automatic* manner, e.g. by supervision of a security expert, in order to avoid the common problem of over or under fitting signatures and to minimize success of mimicry attacks [61]. Thus, the proposed method for signature generation using language models is intended as a tool to support and fasten the process of classical signature generation.

6 Related work and conclusion

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-1960s, character n-grams were used for error correction in optical character recognition [29]. Application of n-grams to text categorization was pioneered by Suen [31] and was followed by a large body of subsequent research [e.g. 26, 27,62]. Various similarity measures were used to compare n-gram frequencies, e.g. the inner product between frequency vectors [27] or Manhattan and Canberra distances [26]. Recent approaches to text categorization advocate the use of kernel functions as similarity measures, which allows one to incorporate contextual information [51,63,64].

Re-discovery of n-gram models in the realm of host-based IDS began in the mid-1990s 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

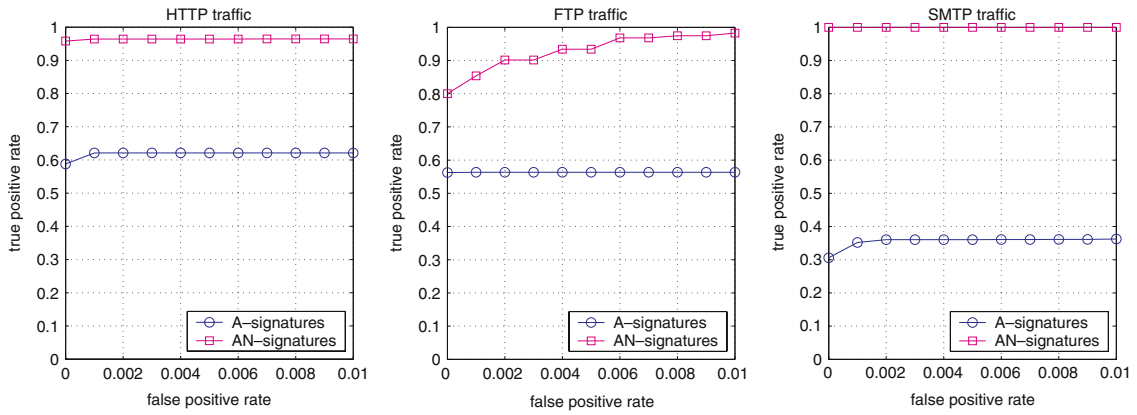


Fig. 10 ROC graphs for A- and AN-signatures (PESIM 2005)

were extended through application of Hidden Markov Models [21], feed-forward and recursive neural networks [23], rule induction algorithms [65] and Support Vector Machines [9]. As part of this evolution, trie and suffix tree data structures were introduced for storage and analysis of system call *n*-grams [22,24,66]. Beside system call analysis, *n*-gram models have recently been applied as part of host-based intrusion detection for identification of malicious code in program binaries and documents [e.g. 67–69].

Application of *n*-gram models for network-based IDS originated in the idea of using byte (1-gram) histograms of packet payloads for statistical tests of abnormality [8,13,18]. A more advanced model was proposed by Wang et al., in which a Bloom filter is applied for storage of high-order *n*-grams of normal and malicious packet payloads [25]. Depending on the ratio of matching normal and anomalous *n*-grams in the filter incoming packets are flagged as either benign or malicious. The approach proposed in this paper differs from these methods as (a) high-order *n*-grams and words of payloads are used to geometrically represent individual connections and (b) unsupervised anomaly detection methods successfully operate without any prior training phase.

Evasion of anomaly detection methods as proposed in [61] is more difficult for high-order *n*-grams and especially variable-length models, since blending and adaptation of attacks requires matching malicious patterns, e.g. assembler instructions in shell-codes, to *consecutive byte sequences* of normal traffic.

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 combined *n*-gram detectors or 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—even though all attack instances were unknown to our system and no prior training was performed.

Beside high detection accuracy, language models support interpretation and explanation of reported anomalies. The presented techniques emphasize significant patterns in anomalies and further accelerate the interlinkage with signature-based security defenses. Thus, unsupervised anomaly detection using language models can be seen as a vital supplement to current security mechanisms by enabling detection and processing of unknown network attacks.

References

1. Staniford, S., Paxson, V., Weaver, N.: How to own the internet in your spare time. In: Proceedings of USENIX Security Symposium (2002)
2. Shannon, C., Moore, D.: The spread of the Witty worm. IEEE Sec. Priv. **2**(4), 46–50 (2004)
3. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer worm. IEEE Sec. Priv. **1**(4), 33–39 (2003)
4. CERT: Advisory CA-2001–21: Buffer overflow in telnetd. CERT Coordination Center (2001)
5. CERT: Advisory CA-2002–28: Openssh vulnerabilities in challenge response handling. CERT Coordination Center (2002)
6. Mahoney, M., Chan, P.: PHAD: packet header anomaly detection for identifying hostile network traffic. Technical Report CS-2001–2, Florida Institute of Technology (2001)
7. Mahoney, V., Chan, K.P.: Learning rules for anomaly detection of hostile network traffic. In: Proceedings of International Conference on Data Mining (ICDM) (2003)

8. Kruegel, C., Toth, T., Kirda, E.: Service specific anomaly detection for network intrusion detection. In: Proceedings of ACM Symposium on Applied Computing, 201–208 (2002)
9. 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, Dordrecht (2002)
10. Mahoney, M., Chan, P.: An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In: Recent Adances in Intrusion Detection (RAID), 220–237 (2004)
11. Vargiya, R., Chan, P.: Boundary detection in tokenizing netwok application payload for anomaly detection. In: Proceedings of ICDM Workshop on Data Mining for Computer Security, 50–59 (2003)
12. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of 10th ACM Conference on Computer and Communications Security, 251–261 (2003)
13. Wang, K., Stolfo, S.: Anomalous payload-based network intrusion detection. In: Recent Adances in Intrusion Detection (RAID), 203–222 (2004)
14. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Trans. Inform. Syst. Sec. **3**, 227–261 (2001)
15. Mahoney, M., Chan, P.: Learning models of network traffic for detecting novel attacks. Technical Report CS-2002–8, Florida Institute of Technology (2002)
16. Mahoney, M.: Network traffic anomaly detection based on packet bytes. In: Proceedings of ACM Symposium on Applied Computing, 346–350 (2003)
17. Zanero, S., Savaresi, S.M.: Unsupervised learning techniques for an intrusion detection system. In: Proceedings of ACM Symposium on Applied Computing (2004)
18. Wang, K., Cretu, G., Stolfo, S.: Anomalous payload-based worm detection and signature generation. In: Recent Adances in Intrusion Detection (RAID) (2005)
19. Forrest, S., Hofmeyr, S., Somayaji, A., Longstaff, T.: A sense of self for unix processes. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, 120–128 (1996)
20. Hofmeyr, S., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. J. Comput. Sec. **6**(3), 151–180 (1998)
21. Warrender, C., Forrest, S., Perlmutter, B.: Detecting intrusions using system calls: alternative data models. In: Proceedings of IEEE Symposium on Security and Privacy 133–145 (1999)
22. Marceau, C.: Characterizing the behavior of a program using multiple-length *n*-grams. In: Proceedings of New Security Paradigms Workshop (NSPW) 101–110 (2000)
23. Ghosh, A., Schwartzbard, A., Schatz, M.: Learning program behavior profiles for intrusion detection. In: Proceedings of USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, 51–62 (1999)
24. Eskin, E., Lee, W., Stolfo, S.: Modeling system calls for intrusion detection with dynamic window sizes. In: Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX) (2001)
25. Wang, K., Parekh, J., Stolfo, S.: Anagram: a content anomaly detector resistant to mimicry attack. In: Recent Adances in Intrusion Detection (RAID) 226–248 (2006)
26. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proceedings SDAIR, Las Vegas 161–175 (1994)
27. Damashek, M.: Gauging similarity with *n*-grams: language-independent categorization of text. Science **267**(5199), 843–848 (1995)
28. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, LS VIII, University of Dortmund (1997)
29. Nagy, G.: Twenty years of document image analysis in PAMI. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 36–62 (2000)
30. Salton, G.: Mathematics and information retrieval. J. Doc. **35**(1), 1–29 (1979)
31. Suen, C.Y.: *N*-gram statistics for natural language understanding and text processing. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2), 164–172 (1979)
32. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proceedings of ACM CSS Workshop on Data Mining Applied to Security (2001)
33. Emran, S., Ye, N.: Robustness of canberra metric in computer intrusion detection. In: Proceedings of IEEE Workshop on Information Assurance and Security, West Point (2001)
34. Jaccard, P.: Contribution au problème de l'immigration post-glaciaire de la flore alpine. Bulletin de la Société Vaudoise Des Sciences Naturelles **36**, 87–130 (1900)
35. Anderberg, M.: Cluster Analysis for Applications. Academic, New York (1973)
36. de la Briandais, R.: File searching using variable length keys. In: Proceedings AFIPS Western Joint Computer Conference 295–298 (1959)
37. Fredkin, E.: Trie memory. Commun. 3(9):490–499: ACM, (1960)
38. Knuth, D.: The art of computer programming, vol. 3. Addison-Wesley, New York (1973)
39. Rieck, K., Laskov, P., Müller, K.R.: Efficient algorithms for similarity measures over sequential data: a look beyond kernels. In: Pattern Recognition, Proceedings of 28th DAGM Symposium. LNCS 374–383 (2006)
40. Rieck, K., Laskov, P., Sonnenburg, S.: Computation of similarity measures for sequential data using generalized suffix trees. In: Advances in Neural Information Processing Systems 19, MIT, Cambridge (2006)
41. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection,. In: Proceedings of SIAM International Conference on Data Mining (2003)
42. Laskov, P., Schäfer, C., Kotenko, I.: Intrusion detection in unlabeled data with quarter–sphere support vector machines. In: Detection of Intrusions and Malware, and Vulnerability Assessment, Proceedings of DIMVA Conference, 71–82 (2004)
43. Laskov, P., Düssel, P., Schäfer, C., Rieck, K.: Learning intrusion detection: supervised or unsupervised? In: Image Analysis and Processing, Proceedings of 13th ICIAP Conference, 50–57 (2005)
44. Rieck, K., Laskov, P.: Detecting unknown network attacks using language models. In: Detection of Intrusions and Malware, and Vulnerability Assessment, Proceedings of 3rd DIMVA Conference. LNCS, 74–90 (2006)
45. Knorr, E., Ng, R., Tucakov, V.: Distance-based outliers: algorithms and applications. Int. J. Very Large Data Bases **8**(3–4), 237–253 (2000)
46. Harmeling, S., Dornhege, G., Tax, D., Meinecke, F.C., Müller, K.R.: From outliers to prototypes: ordering data. Neurocomputing **69**(13–15), 1608–1618 (2006)
47. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Comput. Netw. **34**(4), 579–595 (2000)
48. McHugh, J.: The 1998 Lincoln Laboratory IDS evaluation. In: Recent Adances in Intrusion Detection (RAID) 145–161 (2000)

49. 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. Inform. Syst. Sec. **3**(4), 262–294 (2000)
50. Moore, H.D.: The metasploit project—open-source platform for developing, testing, and using exploit code. http://www.metasploit.com (2005)
51. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. **2**, 419–444 (2002)
52. Tan, K., Maxion, R.: "Why 6?" Defining the operational limits of stide, an anomaly-based intrusion detector. In: Proceedings of IEEE Symposium on Security and Privacy, 188–201 (2002)
53. Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proceedings of USENIX Large Installation System Administration Conference LISA, 229–238 (1999)
54. Microsoft: MS00-078—web server folder traversal vulnerability. Microsoft Sec. Bull. (2000)
55. Anonymous: Once upon a free() ... Phrack Magazine **0xb**(0x39) (2001) 57–0x09
56. Kreibich, C., Crowcroft, J.: Honeycomb—creating intrusion detection signatures using honeypots. In: Proceedings of Workshop on Hot Topics in Networks (2003)
57. Kim, H.A., Karp, B.: Autograph: toward automated, distributed worm signature detection. In: Proceedings of USENIX Security Symposium (2004)
58. Singh, S., Estan, G., Varghese, G., Savage, S.: Automated worm fingerprinting. In: Proceedings of USENIX OSDI (2004)
59. Newsome, J., Karp, B., Song, D.: Polygraph: automatically generating signatures for polymorphic worms. In: Proceedings of IEEE Symposium on Security and Privacy 120–132 (2005)
60. Microsoft: MS05-021—vulnerability in exchange server could allow remote code execution: Microsoft Sec Bull. (2005)
61. Kolesnik, O., Dagon, D., Lee, W.: Advanced polymorphic worms: evading IDS by blending with normal traffic. In: Proceedings of USENIX Security Symposium (2004)
62. Robertson, A.M., Willett, P.: Applications of *n*-grams in textual information systems. J. Doc. **58**(1), 48–69 (1998)
63. Watkins, C.: Dynamic alignment kernels. In: Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., (eds) Advances in large Margin Classifiers, MIT, Cambridge 39–50 (2000)
64. Leslie, C., Eskin, E., Noble, W.: The spectrum kernel: a string kernel for SVM protein classification. In: Proceedings Pacific Symposium Biocomputing. 564–575 (2002)
65. Lee, W., Stolfo, S., Chan, P.: Learning patterns from unix process execution traces for intrusion detection. In: Proceedings of AAAI Workshop on Fraud Detection and Risk Management, Providence 50–56 (1997)
66. Michael, C.: Finding the vocabulary of program behavior data for anomaly detection. In: Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX) 152–163 (2003)
67. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidanm, R.: Detection of new malicious code using *n*-grams signatures. In: Proceedings Second Annual Conference on Privacy, Security and Trust, 193–196 (2004)
68. Karim, M., Walenstein, A., Lakhotia, A., Laxmi, P.: Malware phylogeny generation using permutations of code. J. Comput. Virol. **1**(1–2), 13–23 (2005)
69. Kolter, J., Maloof, M.: Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res. (2006) (to appear)