International Journal of Information Security manuscript No.
(will be inserted by the editor)

Detecting Zero-Day Attacks Using Context-Aware Anomaly Detection At Application-Layer

Patrick Duessel · Christian Gehl · Ulrich Flegel · Sven Dietrich · Michael Meier

Received: date / Accepted: date

Abstract Anomaly detection allows for the identification of unknown and novel attacks in network traffic. However, current approaches for anomaly detection of network packet payloads are limited to the analysis of plain byte sequences. Experiments have shown that application-layer attacks become difficult to detect in the presence of attack obfuscation using payload customization. The ability to incorporate syntactic context into anomaly detection provides valuable information and increases detection accuracy. In this contribution, we address the issue of incorporating protocol context into payload-based anomaly detection. We present a new data representation, called c_n-grams, that allows to integrate syntactic and sequential features of payloads in an unified feature space and provides the basis for context-aware detection of network intrusions. We conduct experiments on both text-based and binary application-layer protocols which demonstrate superior accuracy on the detection of various types of attacks over regular anomaly detection methods. Furthermore, we show how c_n-grams can be used to interpret detected anomalies and thus, provide explainable decisions in practice.

Keywords Intrusion detection · machine learning · anomaly detection · protocol analysis · deep packet inspection

Patrick Duessel
Deloitte & Touche LLP
Cyber Risk Services | Risk Analytics
30 Rockefeller Plaza, New York, NY 10112-0015, United States
Tel.: +1 (212) 492 2332
E-mail: paduessel@deloitte.com

Christian Gehl
Trifense GmbH - Intelligent Network Defense
Germendorfer Strasse 79, 16727 Velten, Germany
Tel.: + 49 (0) 3304 360 368
E-mail: christian.gehl@trifense.de

Ulrich Flegel
Infineon Technologies AG
Am Campeon 1-12, 86579 Neubiberg, Germany
Tel.: + 49 (0) 89 234 21728
E-mail: ulrich.flegel@infineon.com

Sven Dietrich
CUNY John Jay College of Criminal Justice
Mathematics and Computer Science Department
524 W 59th St, 10019 New York, United States
E-mail: spock@ieee.org

Michael Meier
University of Bonn
Friedrich-Ebert-Allee 144, 5313 Bonn, Germany
Tel.: +49 228 73 54249
E-mail: mm@cs.uni-bonn.de

1 Introduction

The analysis of application-layer content in network traffic is getting increasingly important for the protection of complex business environments which deploy a variety of application-specific services and allow for access and transfer of sensitive data between untrusted communication parties. Nowadays, the majority of attacks is carried out at the application-layer. Therefore, monitoring the content of a respective application-layer protocol becomes vital for the detection of unknown and novel application-specific attacks, so called Zero-day attacks.
Signature-based intrusion detection systems (IDS) possess a number of mechanisms for analyzing application-layer protocol content ranging from byte pattern matching as provided by Snort [28] to sophisticated protocol analysis as realized in Bro [22, 23].

However, the major drawback of signature-based IDS is their reliance on appropriate exploit signatures in order to provide adequate protection. Unfortunately, keeping signatures up-to-date is a tedious and resource intensive task given the rapid development of new exploits and their growing variability. This motivates the investigation of alternative techniques.

Payload-based anomaly detection is capable to instantaneously detect previously unknown application-layer attacks. Unlike signature-based systems which search for *explicit* byte patterns, payload-based anomaly detection systems must translate *general knowledge* about patterns into a numeric measure of abnormality which is usually defined by a distance from some model learned over normal payloads.

Thereby, the choice of data representation which is required to measure similarity between sequential data strongly affects the capabilities of an anomaly detector at hand. Sequential data representations, such as n-grams of payload bytes [25, 34], exhibit superior precision at the detection of unknown overflow-based attacks. However, this type of data representation does not adequately account for structural sensitivity needed for detection of rather inconspicuous looking attacks such as cross-site scripting (XSS) or SQL injection. By accessing protocol context of attack patterns significant improvements in the detection accuracy of unknown application-layer attacks can be achieved [5, 9, 14].

In this article, we propose a novel representation of network payloads that integrates protocol context and byte sequences into an unified feature space and thus, allows for a context-aware detection of network intrusions. To this end, a protocol analyzer transforms a network byte stream into the structured representation of a parse tree. Tree nodes are extracted and inserted as tuples of token/attributes into the c_n-gram data structure, a novel data representation of network traffic that allows to efficiently combine sequential models with protocol tokens. Moreover, explainability and the capability to visualize suspicious content with respect to protocol context is another advantage of this data structure over sequential representations.

In order to illustrate the effectiveness of the proposed context-aware payload analysis, we conduct an extensive experimental evaluation in which c_n-grams are compared to conventional n-grams in the presence of a diversity of attacks carrying various payloads. In order to point out the strengths of the proposed data representation, attacks not only include buffer overflows but also web application attacks. Such attacks, for example SQL injections, XSS injections and other script injection attacks, are particularly difficult to detect due to their variability and their strong entanglement in the protocol framework, which makes content analysis based on sequential features ineffective. Our experiments are carried out on network traffic containing text-based and binary application-layer protocols.

The paper is structured as follows: The main contribution of the paper is presented in Section 2 and provides details on a novel data representation for network payloads which allows for the computation of context-aware sequential similarity by geometric anomaly detection methods. A comprehensive experimental evaluation of the proposed method on network traffic featuring various application-layer protocols is carried out in Section 3. Related work on content-based anomaly detection and protocol analysis is presented in Section 4. Finally, conclusions and an outline of future work can be found in Section 5.

2 Methodology

The following four stages outline the essential building blocks of our approach and will be explained in detail for the rest of this section.

1. **Data Acquisition and Normalization**. Inbound packets are captured from the network, reassembled and forwarded to a protocol analyzer such as *binpac* [21] which allows to extract application-layer messages from both text-based and binary protocols. A key benefit of using protocol dissectors as part of data pre-processing is the capability to incorporate expert knowledge in the subsequent feature extraction process. Details on protocol analysis can be found in Section 2.1.
2. **Feature Extraction**. At this stage, byte messages are mapped into a metric space using data representations and features which reflect essential characteristics of a byte sequence. Our approach allows to combine byte-level and syntax-level features in an unified metric space. Details of the feature extraction process can be found in Section 2.2.
3. **Similarity Computation**. The similarity computation between strings is a crucial task for payload-based anomaly detection. With the utilization of vectorial data representations messages can be compared by computing their pairwise distance in the designated geometric space. Similarity measures are explained in Section 2.3.
4. **Anomaly Detection**. In an initial training phase the anomaly detection algorithm learns a global normality model. At detection time a message is compared to the learned data model and based on

its distance an anomaly score is computed. Details on the anomaly detection process can be found in Section 2.4.

2.1 Protocol Analysis

Network protocol analysis is a useful technique to decode and understand data which is encapsulated by an application-specific protocol. Many application-level protocols follow the notion of a common protocol design which is reflected in a unified protocol structure [3]. The majority of application-level protocols stipulate the concept of an application session between two endpoints in which a series of messages is exchanged to accomplish a specific task. Thereby, a *protocol state machine* determines the structure of the application session and specifies legitimate sequences of messages allowed by the protocol. Another essential element of an application protocol is the *message format specification* which defines the structure of an application-layer message. A message format specifies a sequence of fields and their corresponding notion. The syntax of an application-layer protocol can usually be specified by an augmented Backus-Naur-Form which is used to express formal grammars that generate context-free languages. Application protocol analyzer, such as binpac [21], allow to transform re-assembled application-layer messages into a structured data representation, e.g.*parse trees*, which entangle transferred user data with syntactic aspects of the underlying application-layer protocol. In this contribution, we focus on the analysis of message format specifications and do not address the problem of inferring protocol state machines which has been sufficiently addressed in the past. The proposed method is demonstrated using the two application-layer protocols *HTTP* and *RPC* which are explained in detail in the following sections.

2.1.1 Hyper-Text Transfer Protocol

The Hyper-Text Transfer Protocol (HTTP) is one of the most popular text-based application-layer protocols. An example of a typical HTTP request is given below. Control characters are shown as ␣.

```
GET␣/search?q=network+security&gws=ssl&pr=20␣HTTP/1.1..
Host:␣www.google.de..User-Agent:␣Mozilla/5.0␣(X11;␣..
Linux␣x86_64;␣rv:12.0)␣Gecko/20100101␣Firefox/12.0..
Accept:␣text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8..Accept-Language:␣en-us,en;q=0.5..
Accept-Encoding:␣gzip,␣deflate..Connection:␣keep-
alive....
```

The GET request contains CGI parameters as well as common HTTP headers. With the protocol specification at hand a protocol analyzer generates a structured representation of the request sequence which is typically realized by parse trees. An example of a corresponding parse tree is shown in Fig. 1(a). The tree consists of non-terminal nodes as well as pre-terminal and terminal nodes. However, due to the limited complexity of the underlying HTTP grammar, relevant information resides at the pre-terminal and terminal level only. Therefore, the tree can be shrunk and converted into a set of key/attribute tuples. Thereby, each pre-terminal node label serves as a unique protocol context key whereas the associated attribute is assembled from connected terminal nodes.

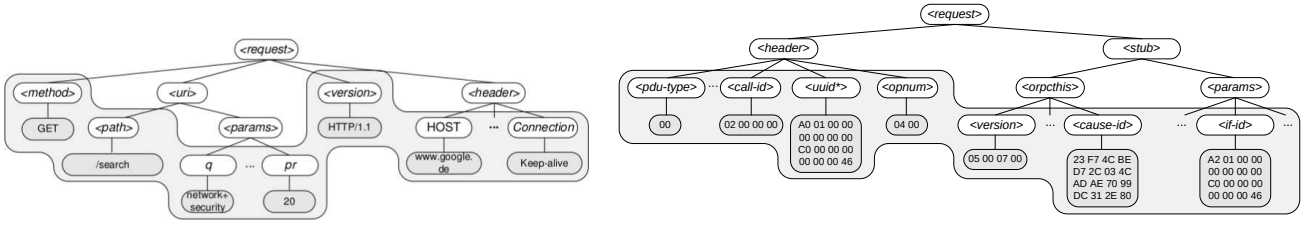
2.1.2 Remote Procedure Calls

A more opaque application-layer protocol is provided by Remote Procedure Call (RPC). A significant part of the Microsoft Windows architecture is composed of services (e.g. DNS, DHCP, DCOM) that communicate with each other in order to accomplish a particular task. Microsoft RPC is a widely used binary application-layer protocol and represents a powerful technology which is utilized by a multitude of services to access functions located at foreign address spaces.
In order to invoke methods remotely, RPC requires to establish a session context. By submitting a `BIND` request the client initiates an RPC session in which the endpoint mapper interface is requested to bind to the desired RPC interface. An example of a `BIND` request is shown below:

```
0000 05 00 0b_A 03 10 00 00 00 78 00 28 00 02 00 00 00
0010 d0 16 d0 16 92 bc 00 00 01 00 00 00 01 00_B 01 00
0020 a0 01 00 00 00 00 00 00 c0 00 00 00 00 00 00 46_C
0030 00 00 00 00 04 5d 88 8a eb 1c c9 11 9f e8 08 00
0040 2b 10 48 60 02 00 00 00_D ...
```

The most important fields of a `BIND` request are highlighted and include protocol data unit type (A) as well as RPC session information. Each session is essentially defined by a context identifier (B) and a universally unique identifier (UUID) which corresponds to the requested RPC interface (C). In order to allow for transfer encoding negotiation, the client provides a coding scheme (D) to the server for each session requested.
The endpoint mapper resolves and returns the endpoint (TCP port) in response to the interface request. Once the client obtains the endpoint it connects to the interface and invokes the desired method by sending a `CALL` request.

```
0000 05 00 00_A 03 10 00 00 00 20 03 00 00 02 00 00 00_E
0010 08 03 00 00 01 00_B 04 00_F 05 00 07 00 01 00 00 00
```



(a) HTTP request parse tree

(b) RPC/DCOM request parse tree (*context identifier is dynamically replaced by the corresponding UUID)

Fig. 1 Generated parse trees representing application-layer protocol requests

```
0020 00 00 00 00 23 f7 4c be d7 2c 03 4c ad ae 70 99
0030 dc 31 2e 80 00 00 00 00 00 00 00 00 00 00 02 00
0040 d8 02 00 00 d8 02 00 00 4d 45 4f 57 04 00 00 00
0050 a2 01 00 00 00 00 00 00 c0 00 00 00 00 00 00 46   G
0060 38 03 00 00 00 00 00 00 c0 00 00 00 00 00 00 46
0070 00 00 00 00 a8 02 00 00 a0 02 00 ...
```

The header essentially specifies a call identifier (E), a session context identifier (B), a method identifier (F) and payload which contains arguments expected by the method. Since the context identifier (B) refers to an active application session in which the client has bound to an interface already the UUID is not explicitly transmitted in a `CALL` request but instead, referenced by the corresponding context identifier.
With the protocol specification at hand the protocol analyzer produces a parse tree which is shown in Fig. 1(b). In this particular example, RPC is used to call the `ISystemActivator` interface in order to request instantiation of a class which is identified by the UUID (G) in the parameter section of the RPC request.
Certainly, method call details can only be extracted from the request if an appropriate RPC stub dissector is in place which is able to analyze RPC payload according to a list of known core interfaces and method declarations. For our considerations, Wireshark's [35] DCE/RPC dissection module is used which allows for concise and automatic parameter value extraction of functions that are declared by well-known RPC interfaces (e.g. LSARPC and SRVSVC).

2.2 Feature Extraction

Application payload is characterized by sequential data which is not applicable for learning methods that operate in metric spaces. Therefore, feature extraction must be performed in order to map sequences into a metric space in which similarity between vectorial representations of sequences can be computed. Formally, a feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^N$ can be defined which maps a data point in the domain of application payloads $\mathcal{X}$ into a N-dimensional metric space - in the following referred to as feature space F - over real numbers:

$$x \longmapsto \phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_N(x)), \tag{1}$$

where $\phi_i(x) \in \mathbb{R}_{\geq 0}$ represents the value of the i-th feature. Thereby, the sole choice of the mapping function $\phi(x)$ provides a powerful instrument to transform data into a representation that is suitable for a given problem.
In this section we describe feature mappings based on different types of features. While protocol analysis suggests to extract features from tree structures such as parse trees, the detection of suspicious byte patterns favors the extraction of sequential features. Once a feature space has been designed, there are several feature embeddings to chose from. Common feature embeddings include binary, count as well as frequency representations of individual features.

Syntax Features. The syntactic structure of transferred application-level payload can be extracted by conducting protocol analysis which eventually allows to generate parse trees. An intuitive way to characterize a parse tree structure is to consider each node independently of its syntactic context, i.e. predecessors as well as successors. The following feature map can be used to determine structural similarity between sequences:

$$\phi : s \longmapsto (\phi_\tau(s))_{\tau \in \mathcal{T}^*} \in F \ ,$$

where $\mathcal{T}^*$ denotes the set of all possible unique subtrees. The mapping function $\phi(s)$ is defined as follows:

$$\phi_t(s) = \begin{cases} 1, & t \in \{\tau \in \mathcal{T}^* \mid n(\tau) = 1\} \\ 0, & otherwise, \end{cases} \tag{2}$$

where $n(\tau)$ is a function which returns the number child nodes attached to a node τ. Using this mapping, each dimension in F corresponds to a binary feature indicating the presence of a particular pre-terminal node t in the actual parse tree of a sequence s. For

the rest of this paper, the set of pre-terminal nodes as a representation of an application-level message is referred to as *bag-of-token* features.

Sequential Features. An intuitive data representation at byte-level involves the extraction of unique substrings by moving a sliding window of length n over a sequence. The resulting set of feature strings are called n-grams. Each sequence s is embedded into a n-dimensional metric space F where $F \in \mathbb{R}^n$, using the following feature map:

$$\phi : s \longmapsto (\phi_w(s))_{w \in \Sigma^n} \in F,$$

where Σ^n refers to the set of all possible strings w of length n induced by an alphabet Σ.

Context-aware Sequential Features. Protocol dissection allows to attach syntactic information to sequential features. By introducing a novel data representation, so called contextual n-grams (c_n-grams), syntactic features can be combined with sequential features in an unified feature space using the feature mapping $\phi(s)$ below.

$$\phi : s \longmapsto (\phi_{w,\tau}(s))_{w_\tau \in \Sigma^n} \in F,$$

where Σ^n refers to the set of all possible strings w_τ of length n induced by an alphabet Σ and $\tau \in \mathcal{T}^*$ refers to a subtree in the set of all possible subtrees. A schematic illustration of c_n-grams is shown in Fig. 2.

The c_n-gram data structure allows to efficiently store n-grams along with syntactic labels. Each entry in the data structure has a unique hash value. The hash value encodes both syntactic context and sequential information represents a c_n-gram. The syntactic label information (i.e. pre-terminal nodes from the parse tree) is encoded using the first k-bits of the CPU's register size m, whereas the remaining $m - k$ bits are used to encode the actual n-gram ($n \leq \lfloor \frac{m-k}{8} \rfloor$) observed in the terminal string attached to a pre-terminal node. As a result, a particular n-gram is allowed to be contained in terminal strings attached to different pre-terminal nodes which represents an extension to the regular definition of n-grams outlined in 2.2.

In the example shown in Fig. 2 extracted HTTP pre-terminal nodes are encoded and combined with n-grams from parsed terminal strings represented as c_n-gram. Finally, the set of c_n-grams is convoluted in a joint histogram H.

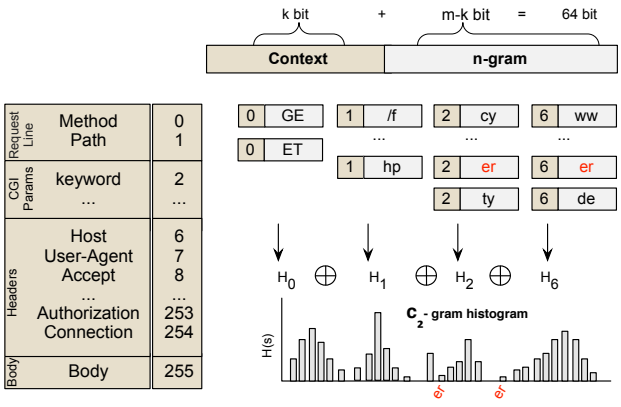


Fig. 2 c_n-grams: context-aware sequential data representation

2.3 Similarity Measure

Once a sequence is mapped into a feature space F a kernel function $k : \mathcal{X}^2 \to \mathbf{R}$ can be applied to determine pairwise similarity between data points $\{x_1, ..., x_n\} \subset \mathcal{X}$. Thereby, the type of kernel function entails an implicit mapping of a data point in F into a possibly even higher dimensional feature space $\mathcal{F}$ which could, in some situations, facilitate the learning process.

In this section we describe two different kernel functions that are most widely used in various application domains, the *Linear Kernel* and the *Radial Basis Function Kernel* (RBF).

Linear Kernel. The linear kernel is defined by a dot product between two vectors x and y and is used to determine similarity between data points which are linearly mapped into $\mathcal{F}$:

$$\begin{aligned} k(x, y) &= \langle \phi(x), \phi(y) \rangle \\ &= \sum_{i=1}^{n} \phi_i(x) \phi_i(y). \end{aligned} \tag{3}$$

With regard to network security, the major benefit of this particular kernel becomes immediately clear. Due to the bijective mapping, a pre-image of every data point in the feature space $\mathcal{F}$ exists which allows to directly deduce differences in features located in F.

Although it seems to quickly become computational unfeasible to compute dot products over sequential features the utilization of efficient data structures such as suffix trees or hash tables allow to compute the similarity $k(x, y)$ in $O(|x| + |y|)$ time [29, 32]. The dot product is of particular mathematical appeal because it provides a geometric interpretation of a similarity score in terms of length of a vector as well as angle and

distance between two vectors. Therefore, the Euclidean distance $d_{eucl}(x, y)$ can be easily derived from the above kernel formulation:

$$\begin{aligned} d_{eucl}(x,y) &= \|x-y\|^2 \\ &= \sqrt{k(x,x)+k(y,y)-2k(x,y)}. \end{aligned} \tag{4}$$

RBF-Kernel. A more complex similarity measure is provided by the RBF-Kernel which implicitly maps data points into a feature space $\mathcal{F}$ which is non-linearly related to the input space. The RBF-kernel is defined as follows:

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \tag{5}$$

where σ controls the width of the gaussian distribution and directly affects the shape of the learner's decision surface. While a large σ results in a linear decision surface which indicates a linearly separable problem, a small value of σ generates a peaky surface which strongly adapts to the distribution of the data in F. The interpretation of RBF-kernel values is non-trivial because, unlike linear kernel functions, an RBF-kernel implicitly maps data points from the input space to an infinite dimensional feature space F. As an example, consider two data points $x, y \in \mathbb{R}^2$, where $x = (x_1, x_2)$, $y = (y_1, y_2)$, the RBF-kernel can be re-formulated as an infinite sum of inner products over features in input space using Taylor series as shown in Eq.(6).

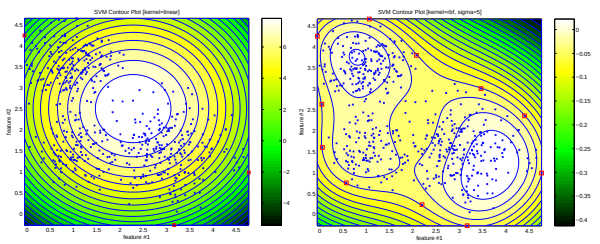
$$\begin{aligned} k(x,y) &= \exp\left(-\|x-y\|^2\right), \\ &= \exp\left(-(x_1-y_1)^2-(x_2-y_2)^2\right), \\ &= \exp\left(-x_1^2+2x_1y_1-y_1^2-x_2^2+2x_2y_2-y_2^2\right), \\ &= \exp\left(-\|x\|^2\right)\cdot\exp\left(-\|y\|^2\right)\cdot\exp\left(2x^Ty\right), \\ &= \exp\left(-\|x\|^2\right)\cdot\exp\left(-\|y\|^2\right)\cdot\textstyle\sum_{n=0}^{\infty}\frac{(2x^Ty)^n}{n!}. \end{aligned} \tag{6}$$

2.4 Anomaly Detection

The problem of anomaly detection can be solved mathematically considering the geometric relationship between vectorial representations of messages. Although anomaly detection methods have been successfully applied to different problems in intrusion detection, e.g. identification of anomalous program behavior [e.g. 7, 8], anomalous packet headers [e.g. 17, 19] or anomalous network payloads [e.g. 12, 25, 26, 33, 34], all methods share the same concept – *anomalies are deviations from a model of normality* – and differ in concrete notions of normality and deviation. For our purpose we use the one-class support vector machine (OC-SVM) proposed in [31] which fits a minimal enclosing hypersphere to the data which is characterized by a center θ and a radius R. Mathematically, this can be formulated as a quadratic programming optimization problem:

$$\begin{aligned} \min_{\substack{R\in\mathbb{R}\\ \xi\in\mathbb{R}^n}} \quad & R^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to:} \quad & ||\phi(x_i)-\theta||^2 \le R^2+\xi_i, \\ & \xi_i \ge 0. \end{aligned} \tag{7}$$

By minimizing R^2 the volume of the hypersphere is minimized given the constraint that training objects are still contained in the sphere which can be expressed by the constraint in Eq.(7).



(a) Linear Kernel (b) RBF Kernel

Fig. 3 Anomaly detection using one-class support vector machine with linear and non-linear decision functions. Support vectors are shown with red edging.

A major benefit of this approach is the control of generalization ability of the algorithm [20], which enables one to cope with noise in the training data and thus dispense with laborious sanitization, as proposed by Cretu et al. [2]. By introducing slack variables ξ_i and penalizing the cost function we allow the constraint to be softened. The regularization parameter $C = \frac{1}{N\nu}$ controls the trade-off between radius and errors (number of training points that violate the constraint) where ν can be interpreted as a permissible fraction of outliers in the training data. The solution of the optimization problem shown in Eq. (7) yields two important facts:

1. The center $\theta = \sum_i \alpha_i\phi(x_i)$ of the sphere can be expressed as a linear combination of training points.
2. Each training point x_i is associated with a weight α_i, $0 \le \alpha_i \le C$, which determines the contribution of the i-th data point to the center and reveals information on the location of x_i. If $\alpha_i = 0$ then

x_i lies in the sphere ($||x_i - c||^2 < R^2$) and the data point can be considered as **normal**. In constrast, if $\alpha_i = C$ x_i can be interpreted as an **outlier** ($||x_i - c||^2 > R^2$). In both cases data points are excluded from the model of "normality". Thus, only those training points yielding $0 < \alpha_i < C$ are located on the surface of the sphere ($||\phi(x_i) - \theta||^2 = R^2$) and thus, define the model of normality as illustrated in Fig. 3. These particular points are known as **support vectors**.

3. The radius R which is explicitly given by the solution of the optimization problem in Eq. (7) refers to the distance from the center θ of the sphere to the boundary (defined by the set of support vectors) and can be interpreted as a threshold for a decision function.

Finally, having determined a model of normality the anomaly score S_z for a test data point z can be defined as the distance from the center in the feature space:

$$\begin{aligned} S_z &= ||\phi(z) - \theta||^2 \\ &= \sum_{w \in \mathcal{A}} (\phi_w(z) - \theta_w)^2 \\ &= \sum_{w \in \mathcal{A}} (\phi_w(z) - \sum_{i=1}^{n} \alpha_i \phi_w(x_i))^2 \\ &= k(z,z) - 2\sum_i \alpha_i k(z, x_i) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j), \end{aligned} \tag{8}$$

where the similarity measure $k(x, y)$ between two points x and y defines a kernel function as introduced in Section 2.3. Depending on the similarity measure at hand data models of different complexity can be learned. For example, as shown in Fig. 3(a) the application of a linear kernel always results in an uniform hypersphere. Thus, the resulting model provides a rather general description of the data. However, if data happens to follow a multi-modal distribution the risk of absorbing outliers in low density regions of the hypersphere might increase. On the contrary, the utilization of an RBF-kernel allows to adopt the distribution characteristics of the data resulting in more complex data models as shown in Fig. 3(b). Of course, the downside of these kind of measures is their lack of interpretability as data points are implicitly mapped into an infinite dimensional feature space in which the identification of individual feature contributions to the overall dissimilarity between two data points becomes difficult (c.f. Section 2.3).

2.5 Feature Visualization

So far, we have discussed how payloads are extracted and mapped into a geometric space in which anomaly detection is carried out to identify deviations from a previously learned model of normality. At this point the following question might arise: why is a data point considered as an anomaly and what constitutes the anomaly? In this section, we derive a feature visualization for payload-based anomaly detection which allows to trace back an anomaly to individual features in the payload and thus, provide a technique that not only helps to understand the reason for an anomaly but also means to localize suspicious pattern. Geometrically, a feature can be considered relevant if it has a significant impact on the norm of a vector. Consequently, the anomaly score $S(z)$ can be expressed as a composition of individual dimensions of $\mathbb{R}^{|\mathcal{A}|}$ as shown in Eq. 8. We refer to $\delta_z = (\phi_w(x) - \theta)^2_{w \in \mathcal{A}}$ as *feature differences*, an intuitive visualization technique to explore sequential disparity which has been originally introduced to determine discriminating q-grams in network traces [25]. The entries of δ_z reflect the individual contribution of a string feature to the deviation from normality represented by θ.

While feature differences provide sufficient means to visualize anomalous network features, a security practitioner might also be interested to directly inspect the portions of the payload that constitute an anomaly. Therefore, the concept of *feature differences* is incorporated into a method known as *feature shading* [27]. The idea of feature shading is to assign a number $m_j \in \mathbf{R}$ to each position j in a payload reflecting its deviation from normality. As a result, the payload can be overlaid with a color shading according to the amount of deviation at a particular position. Considering the generic definition of string features S, each position j in the payload can be associated with multiple feature strings. Hence, a set M_j can be defined which contains all strings s matching a position j of a payload z:

$$M_j = \{z[i, ..., i + |s|] = s \mid s \in S\} \quad j - k + 1 \leq i \leq j \tag{9}$$

where $z[i, ..., i + |s|]$ denotes a substring of length k in z starting at position i . By using M_j the contribution m_j of position j to an anomaly score can be determined as follows:

$$m_j = \frac{1}{|M_j|} \sum_{s \in M_j} - \theta_s^2. \tag{10}$$

An abnormal pattern located at j corresponds to low frequencies in the respective dimensions of the learned model θ and therefore, results in a small value of m_j. A

frequent string is characterized by high values in θ and yields a high value of m_j.

3 Experimental Results

In this section we present results on experiments involving recorded network traffic. Experiments are designed to investigate the benefits of the proposed data representation, specifically with regard to the problem of detecting unknown attacks. Experiments are carried out off-line on two data sets containing two prominent types of network traffic: HTTP and RPC. In our setup, inbound packets are captured, re-assembled and forwarded to *binpac*, a flexible protocol analyzer which extracts and parses application-layer messages. As a baseline, anomaly detection is carried out on application-layer messages using sequential features, i.e. n-grams, and linear as well as non-linear feature mappings. Our experiments are designed to address the following two questions:

1. To which extent does the choice of data representation affect the detection accuracy of an anomaly detector with regard to different network protocols, attack types and basic detector evasion?
2. How does the c_n-gram data representation help to explain and interpret geometric anomalies?

3.1 Data Sets

We evaluate our method on two sanitized data sets containing traffic recorded on a publicly accessible web server (HTTP traffic) as well in an industrial automation testbed (RPC traffic). For both data sets, application-level messages were extracted using the *binpac* protocol parser. Furthermore, we simulated a broad collection of application-level attacks on the target servers and finally mixed those attacks with the captured application-level messages. Attacks were taken from the Metasploit framework [1] as well as from common security forums such as *securityfocus.com*, *remote-exploit.org* and *xssed.com*. Each attack class consists of multiple attack instances carrying different payloads.

3.1.1 ***BLOG09****: Plain-text Protocol - HTTP*

The first data set comprises a sample of 150000 HTTP requests recorded on a blog site web server accessible from the internet over a period of ten days. With an average request length of 389 bytes the size of an HTTP request ranges between 39 and 61127 bytes which is mainly due to the presence of a notable fraction of web spam in the corpus. Protocol analysis reveals 317 distinct tokens including common HTTP protocol elements as well as CGI parameters. In average, each request consists of nine tokens whereas the average length of a corresponding attribute is about 57 bytes. Attacks were chosen to cover a variety of attack types including *buffer overflows* (BUF), *cross site scriptings* (XSS), *SQL injections* (SQL) and *command injections* (CI). Details on the attacks can be found in Table 1. In order to put forward a realistic setup, the majority of attacks was customized to fit existing vulnerabilities. For the rest of the paper this data set is referred to as *BLOG09-I*. Moreover, in order to expose differences in the detection behavior of an anomaly detector based on various data representations we created a second, even larger attack set (*BLOG09-II*) in which the attacks from the original attack set were tuned to increase structural and sequential similarity to normal traffic to impede attack detection. Therefore, HTTP header keys and values frequently observed in normal traffic were added to the attack instances contained in *BLOG09-II*:

```
Host: tim.xxxxxxxx.de..User-Agent: Mozilla/5.0(X11; U
;Linux x86_64; de; rv:1.9.0.7) Gecko/2009030423 Ubunt
u/8.10 (intrepid) Firefox/3.0.7..Accept: text/html,ap
plication/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3.
.Accept-Encoding: gzip,deflate..Accept-Charset: ISO-8
859-1,utf-8;q=0.7,*;q=0.7..Connection: close..
```

Class	CVE	n	Description	Type
1	-	5	wp_profile applet	XSS
2	-	5	wp_profile embed	XSS
3	-	5	wp_profile iframe	XSS
4	-	5	wp_profile body	XSS
5	-	11	wp_profile* crlf	XSS
6	2005-1810	8	wp_index 1=1	SQL
7	2005-1810	5	wp_index outfile	SQL
8	2008-1982	5	wp_index benchmark	SQL
9	2008-4008	11	weblogic trans_enc	BUF
10	2009-0921	9	openview OvOSLocale	BUF
11	2004-1134	8	iis (isapi/w3who)	BUF
12	2001-0241	10	iis (isapi/printing)	BUF
13	2004-0798	10	ipswitch maincfgret	BUF
14	2004-0798	2	edirectory_host	BUF
15	2005-2848	3	barracuda_img_exec	CI

Table 1 HTTP attacks

3.1.2 ***AUT09****: Binary Protocol - RPC*

The second data set consists of 765000 TCP packets that were captured in an industrial automation testbed

[1] http://www.metasploit.com

over a period of eight hours. The traffic contains mostly DCE/RPC traffic. Details on the attack set can be found in Table 2.

Class	CVE	n	Description	Type
1-3	2003-0352	3	RPC/DCOM	BUF
4	2005-0059	1	RPC MSMQ	BUF
5	2007-1748	1	RPC DNS	BUF
6	2003-0533	1	LSASS	BUF
7-11	2006-3439	5	SRVSVC	BUF
12-14	2008-4250	3	SRVSVC	BUF

Table 2 DCE/RPC attacks

3.2 Validation Metrics

Since feature extraction and anomaly detection may require certain parameters to be set our experiments are casted into a cross validation framework in order to determine a parameter configuration which maximizes the detection accuracy for the feature representation in question. To this end, data is split into three distinct partitions for *training*, *validation*, and *testing*. Since we focus on the detection of unknown attacks validation and test partitions contain attacks taken from distinct attack classes. Each model is learned using a sample of 1000 normal messages drawn from the training partition and subsequently validated on 10 distinct validation samples. A validation sample also consists of 1000 normal messages mixed up with attacks from the validation partition. Finally, the model that maximizes the detection accuracy during the validation phase is applied on a test sample. The detection accuracy is measured in terms of area under *receiver operating characteristic* curve ($ROC_{0.01}$) which integrates true positive values over the false positive interval $[0, 0.01]$. For statistical reasons experiments are repeated and results are averaged over 20 repetitions.

3.3 Impact of Data Representation on Detection Accuracy

In this experiment we investigate the impact of different data representations on the detection accuracy with regard to unknown attacks.

BLOG09-I . At first, we present experimental results on the *BLOG09-I* data set. Table 3 shows the model parameters that yield the highest detection accuracy during validation grouped by similarity measure and feature type.

Although the c_n-gram data representation clearly outperforms the bag-of-token features the highest detection rate is achieved by conventional n-grams. The choice of a small n-gram length indicates a comparably modest anomaly detection problem. This is not surprising, since the majority of attacks in the *BLOG09-I* data set is predominantly represented by an attack vector and the respective payload which suggests strong structural as well as sequential dissimilarity to normal traffic. Furthermore, experiments with the RBF-kernel demonstrate a detection behavior which is comparable to the Linear Kernel.

Measure	Feature	n	Embedding	AUC
linear	n-grams	2	bin	0.99 $\pm$0.01
rbf_{10-100}	n-grams	2	bin	0.99 $\pm$0.01
linear	c_n-grams	2	freq	0.92 $\pm$0.01
linear	bag-of-token	-	bin	0.85 $\pm$0.06

Table 3 Area under $\text{ROC}_{0.01}$-curve (AUC) of best models during validation over HTTP requests (*BLOG09-I*) grouped by measure and feature

Given the parameter settings in Table 3 data models are learned and finally applied on separate test samples. The results are depicted in Fig. 4(a) which shows a comparison of ROC-curves that result from the application of different data representations. Similar to the validation results c_n-grams clearly outperform bag-of-token features while n-grams prevail over all data representations. A more detailed analysis of the results is presented in Table 6 which outlines individual false positive rates per attack class (percentage of "normal" test points having an anomaly score above the smallest score achieved by instances of a particular attack class). While the bag-of-token approach suffers false positives around 1% for a few attack classes the detection performance of n-grams and c_n-grams can be considered comparable. However, a major difference in the detection behavior arises for attack class 6 which comprises short and relatively harmless SQL injections of the form "' or 1=1 - -" and which are primarily applied to escalate SQL selection statements.

Results clearly show that the utilization of c_n-grams in that particular case does not provide significant advantages over conventional n-gram data representations. The reason for the poor detection accuracy of that particular attack class resides in the fact, that the amount of suspicious characters make up only a very small contribution to the overall geometric norm of the attack data point embedded

in the c_n-gram feature space which is significantly larger than the conventional n-gram feature space. Therefore, the detection accuracy using the n-gram data representation is significantly higher.

BLOG09-II . We now present results on the *BLOG09-II* data set in which attacks are tuned to match normal traffic characteristics. A list of best models selected during validation is shown in Table 4. In contrast to the validation results on *BLOG09-I* the best model based on n-grams requires a larger string length which suggests a more difficult anomaly detection problem caused by the sole adherence of HTTP protocol headers to the attack vector. As a consequence, the detection accuracy deteriorates for all data representations in comparison to the results of the *BLOG09-I* validation results. As expected, the bag-of-token features completely fail to provide sufficient discriminating information on the attacks yielding a detection accuracy of less than 10% within a false positive interval of [0,0.01]. Interestingly, the data representation providing the highest overall detection accuracy changes for both experiments. While in the *BLOG09-I* experiments the n-gram data representation slightly prevails c_n-grams clearly outperform all other representations in the *BLOG09-II* experiments. Moreover, the utilization of protocol information in combination with sequential features favors the choice of a much smaller string length n while strongly improving the expressiveness of the c_n-gram features.

Measure	Feature	n	Embedding	AUC
linear	c_n-grams	2	freq	0.74 ±0.08
linear	n-grams	4	bin	0.44 ±0.07
rbf_{10-100}	n-grams	4	bin	0.44 ±0.07
linear	bag-of-token	-	bin	0.10 ±0.05

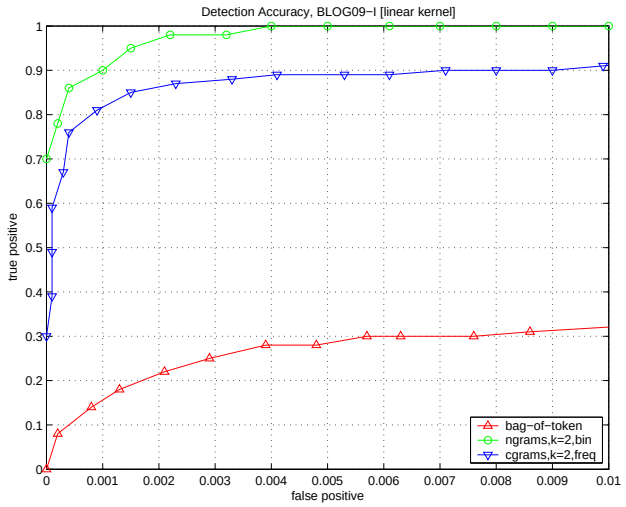
Table 4 Area under $ROC_{0.01}$-curve (*AUC*) of best models during validation over HTTP requests (*BLOG09-II*) grouped by measure and feature

A comparison of ROC-curves for the *BLOG09-II* test data set is depicted in Fig. 4(b). The corresponding false positive rates per attack class are provided in Table 7. While most of the buffer overflows (classes 9-14) can be reliably detected by both n-grams and c_n-grams the utilization of the c_n-gram data representation results in a superior detection accuracy for cross site scripting attacks (classes 1-5) and SQL injections (classes 7-8) which, on the other hand, induces a significant increase in false positives for n-grams. As shown in Fig. 4(b) the incorporation of protocol information into sequential features results in a detection rate of 80% at a false positive level of 1% which is almost twice as high as attained by conventional n-grams (43%).The false positive analysis reveals that by using c_n-grams the accuracy can be strongly improved for majority of non-overflow attacks which is reflected by a major reduction of the respective false positive rate to a level less than 1%.

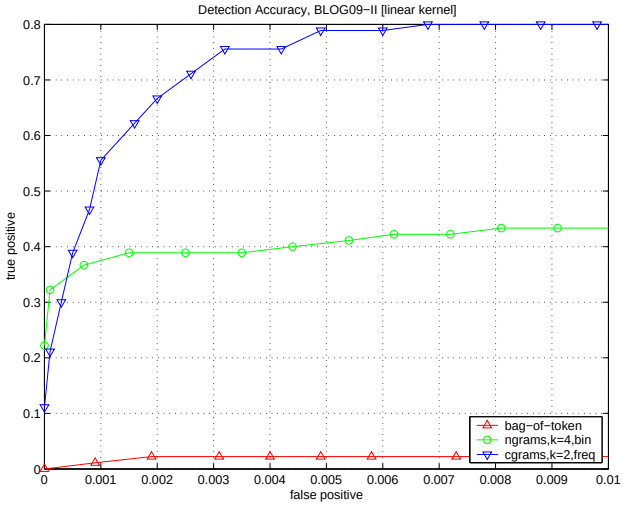
The reason for c_n-grams to outperform conventional n-grams becomes obvious by picturing document frequency differences between normal data and individual attack instances which is shown in Fig 5. The plot displays 1-gram frequency differences between 10000 normal HTTP requests and two different attacks, a buffer overflow (class 14) and a SQL union injection (class 8). A frequency difference of 1 arises from bytes that solely appear in normal data points whereas bytes that are exclusively found in the attack yield a frequency difference of -1. Bytes with a frequency difference close to zero do not provide discriminating information. In the upper two plots frequency differences of both attacks are shown that result from conventional analysis at request-level, whereas the lower two plots show frequency differences of both attacks from token-level analysis. The boxes in the plots mark areas with the largest amount of differences between frequency difference distributions attained from request-level and token-level analysis.

The buffer overflow can be easily detected due to the presence of a large amount of non-printable characters (a large number of points at -1 for byte values in the range of [0,45] and [128,255]). On the other hand, the SQL injection contains mostly ASCII characters, which is reflected by close to zero frequency differences for most of the printable characters. As a consequence, the detection of this kind of attack is comparably difficult using conventional n-gram features. On the contrary, the lower two charts show frequency differences for the same attacks but with regard to the respective vulnerability. The frequency differences of the buffer overflow become even more obvious considering the local byte distribution in the scope of the exploited parameter "Host". Similar clarification takes place for the SQL injection for which many previously normal bytes become clearly anomalous taking the respective vulnerable protocol element ("cat") into account. This results in a major improvement of the detection accuracy.

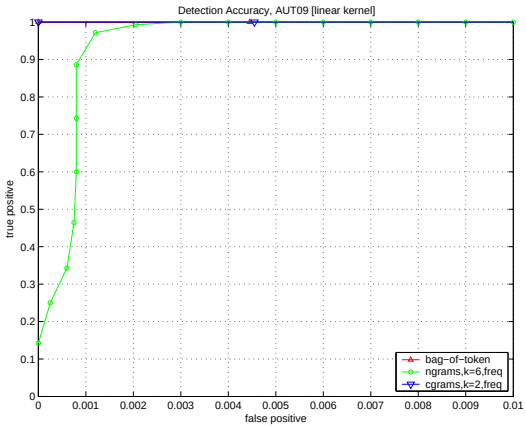
In contrast to the results on the *BLOG09-I* data set which demonstrate insignificant advantages of c_n-grams over conventional n-grams (with regard to the problem of anomaly detection) experiments on the *BLOG09-II* data set show that the utilization of c_n-



(a) HTTP attack set I



(b) HTTP attack set II



(c) RPC attack set

Fig. 4 $ROC_{0.01}$-curve for attacks over binary and text-based application-layer protocols

grams results in a major improvement of detection accuracy, predominantly for web application attacks such as SQL injections and XSS attacks. This is mainly due to the fact that both attack types employ portions of the alphabet that are common with normal traffic. However, since buffer overflows exhibit some kind of binary shell code, the detection accuracy remains largely unaffected for those type of attacks. The results also suggest that in presence of payload customization (e.g. adding frequently occurring HTTP headers) the c_n-gram data representation is more effective than n-grams because the protocol context of a particular n-gram becomes increasingly important if normal and outlier classes are generated by similar alphabets and distributions.

AUT09 . We now present results of experiments conducted on binary network traffic. Table 5 shows the model parameters that yield the highest detection accuracy during validation grouped by similarity measure and feature type. Since multiple best parameter configurations are selected final experiments are carried out using parameters that yield the least complex data model. The results on the test data set are depicted in Fig. 4(c). In contrast to the experiments on HTTP, RPC attacks are perfectly detected using both bag-of-token features and c_n-grams. However, as shown in Table 8 the utilization of c_n-grams demonstrates only a marginal improvement in the detection accuracy compared to n-grams.

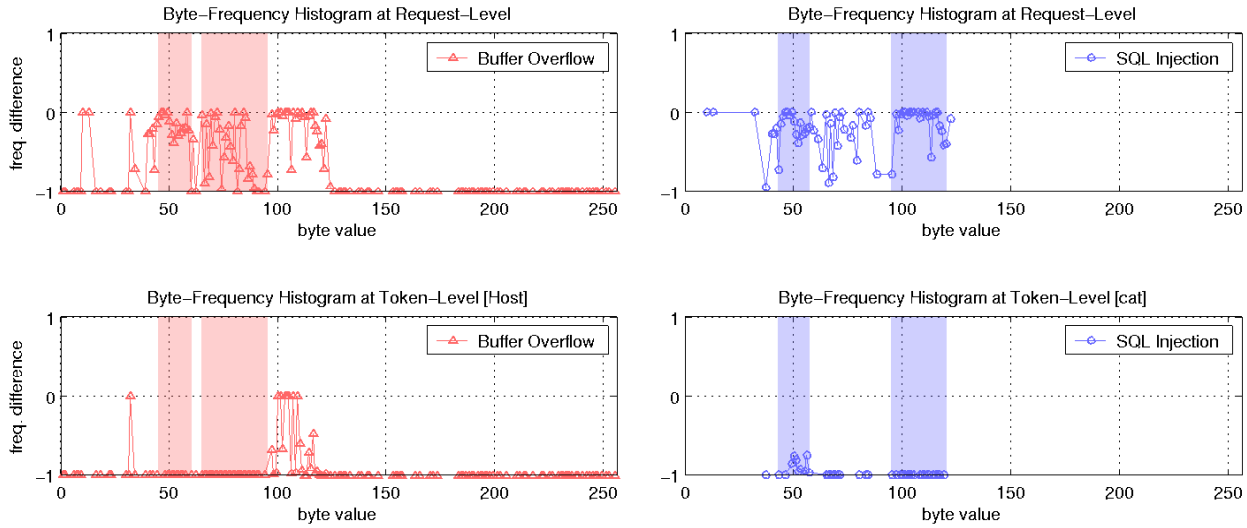


Fig. 5 Byte-frequency differences at request and token level over *BLOG09-II* requests

Measure	Feature	n	Embed	AUC
linear	bag-of-token	-	bin	1
linear	c$_n$-grams	2-4	bin/freq	1
linear	n-grams	4-6	freq	0.99 ±0.01
rbf$_{10-100}$	n-grams	4-6	freq	0.99 ±0.01

Table 5 Area under ROC$_{0.01}$-curve (*AUC*) of best models during validation over RPC requests (AUT09) grouped by measure and feature

3.4 Localization of Anomalous Payloads Using Syntax-Sensitive Features

The c$_n$-gram data representation is not only effective for the detection of web application attacks in network traffic, it can also be used to pinpoint vulnerabilities in protocols and web applications. In this section we demonstrate the use of feature shading, an intuitive method to explain geometric anomalies based on feature differences (c.f. Section 2.5) in presence of protocol context information.

The c$_n$-grams data representation not only allows to track down suspicious byte patterns, it also provides a security operator with information on the syntactic context, e.g. a potentially vulnerable protocol element such as an HTTP header or a CGI parameter. This constitutes a major benefit over conventional n-grams. Geometrically, a protocol element can be considered potentially vulnerable if its associated attribute consists of n-grams which contribute to a large extent to the norm of the overall feature differences vector. An example for a SQL injection is shown in Fig. 6 which displays the contribution to the overall norm as a function of individual protocol elements. The CGI parameter "cat" clearly exhibits the largest fraction of the overall feature differences norm and therefore, might provide substantial information on the attack vector.

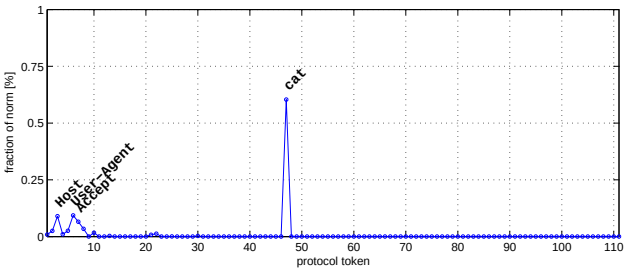


Fig. 6 Contribution of protocol tokens to overall norm of the feature differences vector between 10000 normal HTTP requests and a SQL injection

An example of a corresponding feature shading is given in Fig. 7. For each byte position the corresponding term frequency difference is calculated and graphically highlighted using some coloring scheme. Darker regions exhibit strong frequency differences and thereby, indicate unusual byte patterns. As expected, the most suspicious and longest contiguous byte pattern is located in the attribute of the CGI parameter "cat". In this particular example the attacker mounts a SQL union injection with the objective to create a file on the web server that takes a URI as an argument to eventually allow for remote file inclusion.

The identification of anomalous payloads transferred over HTTP is not very difficult due to the nature of text-based protocols. However, the localization

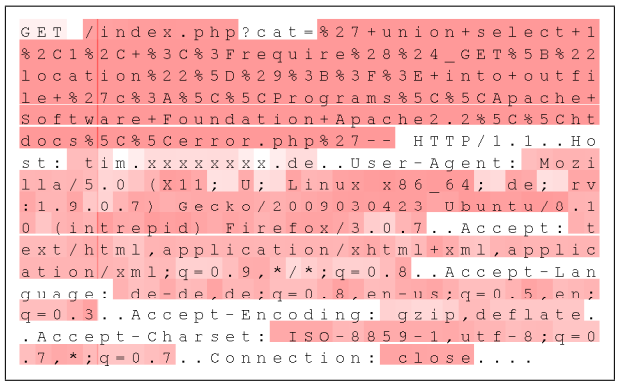


Fig. 7 Feature shading of a SQL union injection based on c_n-grams

of malicious byte patterns transferred over binary protocols is more challenging because the structure of the underlying protocol is not always obvious and typical shell code bytes can also be found in normal traffic. Below is an example for the identification of a potential vulnerability exploited by a RPC buffer overflow.

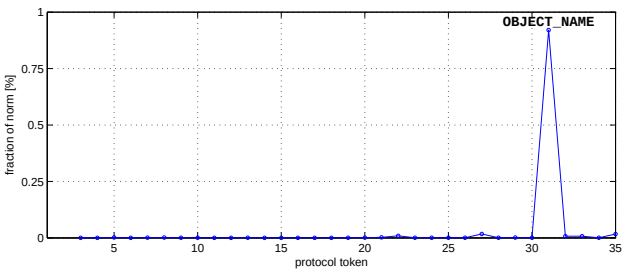


Fig. 8 Contribution of protocol tokens to overall feature differences norm between 10000 normal RPC requests and a RPC buffer overflow

With 93% the largest fraction of the overall feature differences norm is caused by the attribute associated with the token `Object_Name`. The feature shading is presented in Fig. 9 and clearly reveals a long, contiguous byte sequence which according to the opaque coloring suggests anomalous content.

Examination of the corresponding parse tree shows that the attacker binds to the Microsoft Windows remote activation interface registered by the UUID **4d9f4ab8-7d1c-11cf-861e-0020af6e7c57** and exploits a vulnerability located in the *RemoteActivation* function. Thereby, the protocol token `Object_Name` which clearly stands out in Fig. 8 corresponds to a parameter of the *RemoteActivation* function which is overflown by the attacker. Needless to say that the attack vector fits

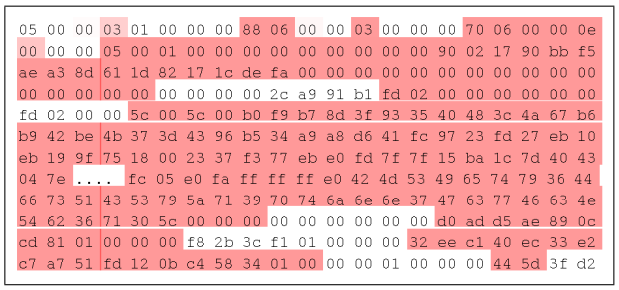


Fig. 9 Feature shading of a RPC buffer overflow attack based on c_n-grams

the well-known RPC/DCOM vulnerability (CVE 2003-0352).

As demonstrated in this section, protocol analysis along with payload-based anomaly detection allows not only to identify suspicious byte patterns in application-layer messages, it also provides a better understanding of attacks and helps to pinpoint potential vulnerabilities.

4 Related Work

Payload-based Anomaly Detection. Over the last decade, a multitude of anomaly-based intrusion detection systems have been proposed which analyze the payload of a packet with regard to sequential anomalies. Wang et al. proposed *Payl* [33] which constructs a simple packet-length specific model of normal traffic. The model relies on the computation of n-gram frequencies ($n \leq 2$) in a payload. The anomaly score is defined by the simplfied Mahalanobis distance to the previously learned n-gram distribution. In a continuitive work, the authors presented *Anagram* [34] which computes a model of normal as well as malicious traffic based on distinct binarized n-grams stored in different Bloom filters. Packets are scored by counting the number of unobserved normal n-grams as well as observed malicious n-grams. An extensive experimental evaluation of geometric outlier detection is provided by Rieck and Laskov [26] who investigated global and local outlier detection methods with regard to the problem of unknown attack detection. Perdisci et al. proposed *McPad* [24] which relies on an ensemble of One-Class SVM detectors. Their approach allows to incorporate different types of features by combining individual detectors in order to achieve a better trade-off between detection accuracy and false positive rate.

Feature Extraction. A large amount of previous work in the domain of network intrusion detection systems has focused on features derived from network and transport layer protocols. An example of such features can be

found in the data mining approach of Lee and Stolfo [15], containing packet, connection and time window features derived from IP and TCP headers. The same work has pioneered the use of "content" features that comprised selected application-level properties such as number of shell prompts, number of failed login prompts, etc. deemed to be relevant for detection of specific attacks. Similar features comprising selected keywords from application-layer protocols have been used by Mahoney and Chan for anomaly detection [18].

General content-based features using the payload distribution of specific byte groups have been first proposed by Kruegel et al. [13] in the context of service-specific anomaly detection using separate normality models for different application-layer protocols. Full distributions of byte values have been considered by Wang and Stolfo [33], eventually extended to models of various languages that can be defined over byte sequences, e.g. n-grams [25, 34]. However, the problem of modeling n-grams is ill-posed because it entails the estimation of distributions in exponentially growing sample spaces. To overcome this problem, Perdisci et al. [24] suggested to use gappy n-grams [16] which effectively reduces the sample space for higher-order n-grams ($n > 2$). Instead of explicitly modeling n-grams, the *Spectrogram* detector [30] which was specifically designed to protect web applications against malicious user input learns a probabilistic representation of legitimate web-layer input using a mixture of factorized n-gram Markov models. Their approach casts n-gram observations into products of conditional probabilities representing transitions between individual characters which eventually results in a linearization of the space complexity.

The incorporation of protocol features into the detection process has been first realized in signature-based IDS. Robust and efficient protocol parsers have been developed for the Bro IDS [22]; however, until recently they were tightly coupled with Bro's signature engine, which has prevented their use in other systems. The development of a stand-alone and extendable protocol parser binpac [21] has provided a possibility for combining protocol analysis with other detection techniques. Incremental and bi-directional parsing as well as error recovery are especially attractive features of binpac. Similar properties at a more abstract level are exhibited by the protocol analyzer GAPAL [1]. Although, protocol parsing by definition limits the scope of analysis to known network protocols unknown protocols can be explored using sophisticated protocol reverse engineering techniques [3, 36].

However, the incorporation of protocol analysis into anomaly detection remains largely unexplored. Kruegel and Vigna [12] have developed a highly effective system for the detection of web attacks by considering separate models for HTTP requests. The system combines models built over specific features such as length, character distribution and request token order defined for individual web applications associated with particular URI paths. Duessel et al. [5] proposed a method which allows to integrate protocol analysis in payload-based anomaly detection based on composite kernel measures. Their experiments showed that significant improvements in the detection accuracy can be achieved, especially for unknown web application attacks. Our method advances this approach in that it allows to transparently map byte sequences into a unique geometric space which reflects sequential as well as syntactical features. Hence, instead of calculating multiple kernels the c_n-gram data structure eventually requires to compute one kernel only.

Evasion. Attack obfuscation is a common practice among attackers to avoid detection. *Blending-based* evasion of byte frequency-based network anomaly IDS has been first addressed by Kolesnikov and Dagon [11] who suggest to blend malicious attack packets with normal traffic using advanced obfuscation techniques such as spectrum analysis [4]. Although Fogla et al. [6] present a proof of NP-hardness of the problem a comprehensive experimental evaluation reveals practical feasibility of blending-based attacks in a continuative study. Resistance of payload-based online anomaly detectors against *poisoning-based* evasion schemes has been investigated by Kloft and Laskov [10].

5 Conclusion

In this contribution we propose a general method that facilitates the combination of protocol analysis and payload-based anomaly detection. To this end, we present a novel data representation, so called c_n-grams, that allows to integrate protocol features and sequential features in an unified geometric feature space.

We conduct extensive experiments on recorded network traffic featuring both text-based and binary application-layer protocols which demonstrate superior accuracy on the detection of unknown attacks. Our proposition shows that novel attacks can be identified reliably and the detection accuracy can be boosted from 44% using unsupervised anomaly detection with plain sequential features to 80% using combined features in the presence of attack obfuscation. Although the additional effort of protocol analysis does not seem to pay off for simple SQL injections the method

proves to be especially useful for the detection of web application attacks such as XSS and SQL injections in the presence of attack obfuscation by payload customization. Moreover, we show how c_n-grams can be used to explain geometric anomalies to security experts and also provide insight into vulnerabilities by identifying and pinpointing meaningful features in a payload stream using the feature shading technique.

Due to its general nature, the proposed feature extraction method can be applied on any protocol for which a protocol analyzer is available. While in this contribution we focus on the utilization of protocol features derived from message format specifications, future work should address the problem of how to incorporate protocol state machines into sequential feature representations and investigate the identified limitation of both data representations regarding the accurate detection of local anomalies in sparse feature spaces.

References

1. N. Borisov, D.J. Brumley, H. Wang, J. Dunagan, P. Joshi, and C. Guo. Generic application-level protocol analyzer and its language. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2007.
2. G. Cretu, A. Stavrou, M. Locasto, S.J. Stolfo, and A.D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *ieeesp*, 2008.
3. W. Cui, J. Kannan, and H.J. Wang. Discoverer: automatic protocol reverse engineering from network traces. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–14, 2007.
4. T. Detristan, T. Ulenspiegel, Y. Malcom, and M. Underduk. Polymorphic shellcode engine using spectrum analysis. Phrack Issue 0x3d, 2003.
5. P. Düssel, C. Gehl, P. Laskov, and K. Rieck. Incorporation of application layer protocol syntax into anomaly detection. In *ICISS*, pages 188–202, 2008.
6. P. Folga, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee. Polymorphic blending attacks. In *Proc. of USENIX Security Symposium*, pages 241–256, 2006.
7. S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff. A sense of self for unix processes. In *Proc. of IEEE Symposium on Security and Privacy*, pages 120–128, Oakland, CA, USA, 1996.
8. D. Gao, M.K. Reiter, and D. Song. Behavioral distance measurement using hidden markov models. In *Recent Adances in Intrusion Detection (RAID)*, pages 19–40, 2006.
9. K.L. Ingham, A. Somayaji, J. Burge, and S. Forrest. Learning DFA representations of HTTP for protecting web applications. *Computer Networks*, 51(5):1239–1255, 2007.
10. M. Kloft and P. Laskov. Security analysis of online centroid anomaly detection. Technical Report UCB/EECS-2010-22, EECS Department, University of California, Berkeley, Feb 2010.
11. O. Kolesnikov, D. Dagon, and W. Lee. Advanced polymorphic worms: Evading IDS by blending with normal traffic. In *Proc. of USENIX Security Symposium*, 2004.
12. C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proc. of 10th ACM Conf. on Computer and Communications Security*, pages 251–261, 2003.
13. C. Kruegel, T. Toth, and E. Kirda. Service specific anomaly detection for network intrusion detection. In *Proc. of ACM Symposium on Applied Computing*, pages 201–208, 2002.
14. C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 48(5), 2005.
15. W. Lee and S.J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information Systems Security*, 3:227–261, 2000.
16. H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.
17. M.V. Mahoney and P.K. Chan. PHAD: Packet header anomaly detection for identifying hostile network traffic. Technical Report CS-2001-2, Florida Institute of Technology, 2001.
18. M.V. Mahoney and P.K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 376–385, 2002.
19. M.V. Mahoney and P.K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Recent Adances in Intrusion Detection (RAID)*, pages 220–237, 2004.
20. K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12(2):181–201, May 2001.
21. R. Pang, V. Paxson, R. Sommer, and L.L. Peterson. binpac: a yacc for writing application protocol parsers. In *Proc. of ACM Internet Measurement Conference*, pages 289–300, 2006.
22. V. Paxson. Bro: a system for detecting network intruders in real-time. In *Proc. of USENIX Security Symposium*, pages 31–51, 1998.
23. V. Paxson. The bro 0.8 user manual. Lawrence Berkeley National Laboratroy and ICSI Center for Internet Research, 2004.
24. R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, In Press, Corrected Proof:–, 2008.
25. K. Rieck and P. Laskov. Detecting unknown network attacks using language models. In *Detection of Intrusions and Malware, and Vulnerability Assessment, Proc. of 3rd DIMVA Conference*, LNCS, pages 74–90, July 2006.
26. K. Rieck and P. Laskov. Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256, 2007.
27. K. Rieck and P. Laskov. Visualization and explanation of payload-based anomaly detection. In *Proc. of European Conference on Computer Network Defense (EC2ND)*, 2009.
28. M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proc. of USENIX Large Installation System Administration Conference LISA*, pages 229–238, 1999.
29. J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.
30. Y. Song, A.D. Keromytis, and S.J. Stolfo. Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2009.

31. D. Tax and R. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proc. ESANN*, pages 251–256, Brussels, 1999. D. Facto Press.
32. S.V.N. Vishwanathan and A.J. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J.F. Vert, editors, *Kernels and Bioinformatics*, pages 113–130. MIT Press, 2004.
33. K. Wang and S.J. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Adances in Intrusion Detection (RAID)*, pages 203–222, 2004.
34. K. Wang, J.J. Parekh, and S.J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Adances in Intrusion Detection (RAID)*, pages 226–248, 2006.
35. Wireshark. Wireshark: Network protocol analyzer. Internet: http://www.wireshark.org, Accessed: 2010.
36. G. Wondracek, P. Milani Comparetti, C. Kruegel, and E. Kirda. Automatic Network Protocol Analysis. In *15th Symposium on Network and Distributed System Security (NDSS)*, 2008.

A Experiment Results Details

Type	Attack Class	**linear**, bag-of-token FP_{avg}	FP_{std}	**linear**, n-**grams** FP_{avg}	FP_{std}	**rbf**, n-**grams** FP_{avg}	FP_{std}	**linear**, cn-**grams** FP_{avg}	FP_{std}
XSS	1	0.0187	±0.0077	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	2	0.0137	±0.0074	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	3	0.0140	±0.0042	0.0000	±0.0000	0.0000	±0.0000	0.0001	±0.0004
	4	0.0150	±0.0070	0.0007	±0.0008	0.0007	±0.0008	0.0005	±0.0012
	5	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
SQL	6	0.0152	±0.0053	0.0027	±0.0015	0.0027	±0.0015	0.6820	±0.0478
	7	0.0118	±0.0015	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	8	0.0147	±0.0057	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
BUF	9	0.0150	±0.0072	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	10	0.0010	±0.0007	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	11	0.0139	±0.0068	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	12	0.0003	±0.0005	0.0000	±0.0000	0.0000	±0.0000	0.0001	±0.0004
	13	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	14	0.1698	±0.0438	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
CI	15	0.0037	±0.0022	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000

Table 6 False positive rates for unknown HTTP attacks (*BLOG09-I*) on test data

Type	Attack Class	**linear**, bag-of-token FP_{avg}	FP_{std}	**linear**, n-**grams** FP_{avg}	FP_{std}	**rbf**, n-**grams** FP_{avg}	FP_{std}	**linear**, cn-**grams** FP_{avg}	FP_{std}
XSS	1	0.4044	±0.0858	0.0802	±0.0301	0.0798	±0.0316	0.0004	±0.0005
	2	0.4250	±0.0000	0.1260	±0.0000	0.1260	±0.0000	0.0000	±0.0000
	3	0.4248	±0.0686	0.2683	±0.0523	0.2682	±0.0511	0.0017	±0.0014
	4	0.3750	±0.0516	0.1223	±0.0154	0.1220	±0.0166	0.0007	±0.0006
	5	0.4452	±0.0637	0.2074	±0.0502	0.2078	±0.0505	0.0012	±0.0008
SQL	6	0.4357	±0.0796	0.4143	±0.0291	0.4143	±0.0289	0.9263	±0.0159
	7	0.6779	±0.1752	0.1333	±0.0356	0.1330	±0.0358	0.0034	±0.0020
	8	0.4697	±0.1048	0.0389	±0.0160	0.0386	±0.0158	0.0019	±0.0012
BUF	9	0.4113	±0.0660	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	10	0.1677	±0.0347	0.0000	±0.0000	0.0000	±0.0000	0.0000	±0.0000
	11	0.4054	±0.0489	0.0000	±0.0000	0.0000	±0.0000	0.0001	±0.0004
	12	0.1274	±0.0728	0.0000	±0.0000	0.0002	±0.0004	0.0012	±0.0008
	13	0.0005	±0.0007	0.0000	±0.0000	0.0000	±0.0000	0.0005	±0.0007
	14	0.8547	±0.0412	0.0000	±0.0000	0.0000	±0.0000	0.0001	±0.0004
CI	15	0.5048	±0.0758	0.3335	±0.0784	0.3318	±0.0757	0.0008	±0.0010

Table 7 Average false positive rates for unknown HTTP attacks (*BLOG09-II*) on test data

Attack Class	**linear**, bag-of-token FP_{avg}	FP_{std}	**linear**, n-**grams** FP_{avg}	FP_{std}	**rbf**, n-**grams** FP_{avg}	FP_{std}	**linear**, c$_n$-**grams** FP_{avg}	FP_{std}
1	0	0	0.0010	±0.0012	0.0010	±0.0012	0	0
2	0	0	0.0012	±0.0010	0.0012	±0.0010	0	0
3	0	0	0.0008	±0.0008	0.0008	±0.0008	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0.0008	±0.0009	0.0001	±0.0003	0	0
8	0	0	0.0008	±0.0010	0.0008	±0.0009	0	0
9	0	0	0.0008	±0.0007	0.0009	±0.0011	0	0
10	0	0	0.0006	±0.0005	0.0006	±0.0005	0	0
11	0	0	0.0008	±0.0008	0.0008	±0.0008	0	0
12	0	0	0.0006	±0.0007	0.0006	±0.0007	0	0
13	0	0	0.0009	±0.0008	0.0009	±0.0008	0	0
14	0	0	0.0004	±0.0005	0.0004	±0.0005	0	0

Table 8 False positive rates for unknown RPC attacks (AUT09) on test data