Future Internet **2010**, *2*, 662-669; doi:10.3390/fi2040662

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

Improving Anomaly Detection for Text-Based Protocols by Exploiting Message Structures

Martin Güthle [1], Jochen Kögel [1,*], Stefan Wahl [2], Matthias Kaschub [1] and Christian M. Mueller [1]

[1] Institute of Communication Networks and Computer Engineering (IKR), University of Stuttgart, Stuttgart, Germany; E-Mails: mguethle@xunit.de (M.G.); matthias.kaschub@ikr.uni-stuttgart.de (M.K.); christian.mueller@ikr.uni-stuttgart.de (C.M.M.)

[2] Bell-Labs Germany, Alcatel-Lucent Deutschland AG, Stuttgart, Germany; E-Mail: Stefan.Wahl@alcatel-lucent.com

* Author to whom correspondence should be addressed; E-Mail: jochen.koegel@ikr.uni-stuttgart.de; Tel.: +49-711-685-69012.

Received: 23 October 2010; in revised form: 16 December 2010 / Accepted: 17 December 2010 / Published: 21 December 2010

Abstract: Service platforms using text-based protocols need to be protected against attacks. Machine-learning algorithms with pattern matching can be used to detect even previously unknown attacks. In this paper, we present an extension to known Support Vector Machine (SVM) based anomaly detection algorithms for the Session Initiation Protocol (SIP). Our contribution is to extend the amount of different features used for classification (feature space) by exploiting the structure of SIP messages, which reduces the false positive rate. Additionally, we show how combining our approach with attribute reduction significantly improves throughput.

Keywords: anomaly detection; classification; text-based protocols; SIP; SVM

1. Introduction

The world of telecommunication is evolving from closed legacy networks towards open IP-based networks. The control plane of these networks mainly relies on text-based protocols, such as the Session Initiation Protocol (SIP) [1]. This protocol is also increasingly used to embed conversational services into numerous new applications for end users *across* multiple mobile and fixed access networks. Therefore, the risk is also increasing that attackers exploit vulnerabilities in SIP processing network elements for gaining control over them or performing fraud attacks.

If an attacker gains control of a network device or somehow disturbs network operation, this quickly results in loss of revenue for the service provider, damages the provider's image, and leads to customer dissatisfaction. For the protection of SIP-based networks, a first approach is deploying firewalls at the edge that operate on a reactive principle using a database containing all known attack patterns. However, attacks for which no patterns are contained in the database, so-called zero day exploits, cannot be detected. A proactive capability to detect previously unknown attack variants can be provided by machine learning algorithms, which capture characteristics of observed normal and/or incorrect messages to classify anomalies as malicious events.

Nassar *et al.* [2] proposed Support Vector Machines (SVM) to detect anomalies in SIP signaling traffic. SVMs are a relatively new approach for classification in high dimensional feature spaces. Instead of reducing the dimensions of the feature space directly, SVMs extend the dimensions even further and separate the classes with a suitable yet simple description of a border in between, called a hyperplane [3]. In contrast to other classification algorithms, the hyperplane does not need to fully separate the classes from each other. Additional support vectors are used to describe exceptions from the hyperplane classes. The hyperplane and support vectors are calculated in the training phase and constitute the classification model. The computational complexity during classification, and thus the achievable throughput, depends on the formulation of this model. While Nassar *et al.* proposed to use *traffic* characteristics as features for the classification, the approach by Rieck *et al.* [4] is based on n-grams that are generated from the SIP *messages*.

As the SIP protocol incorporates many extensions, the false positive rate (FPR) of currently known algorithms is likely to exceed a threshold which endangers the applicability in products. We propose to add protocol knowledge to the feature extraction step to achieve better classification results and herewith lower FPR. We detail our approach in the next section, present an evaluation in Section 3, and summarize the results in Section 4.

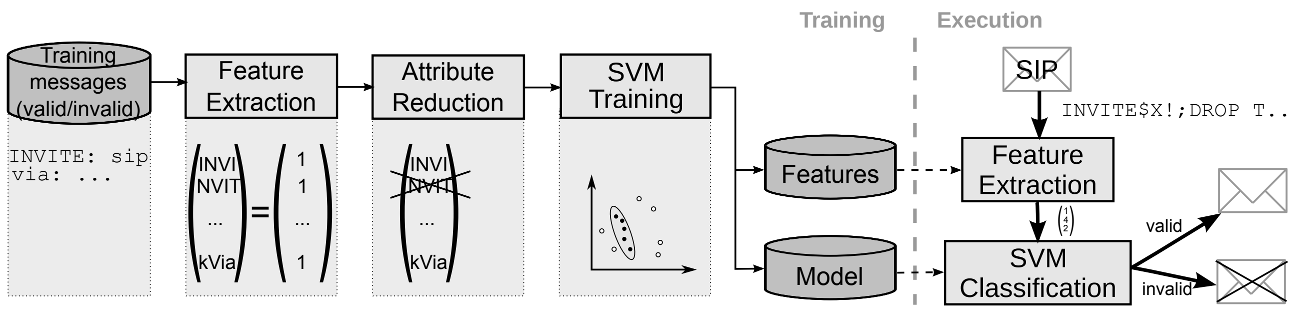
2. Approach and Improvements

In the following we give an overview of our approach, suitable for SIP and other text-based protocols. Then we highlight our extensions for mapping text messages into feature space and how we performed attribute reduction.

2.1. Overview

Deployed at the border of a service platform, the system should decide whether messages are valid or invalid (*i.e.*, suspicious and indicate an attack). Figure 1 shows on the right hand side how the system takes decisions on real messages received from the network, which represents the execution phase. Since the system processes each message independently of other messages and does rely on traffic characteristics, it does not need to keep transaction state. This enables parallelization and makes the system insusceptible to denial-of-service attacks.

Figure 1. Classifier-based anomaly detection system with training and execution phase.



Before execution, the system must be fed with training data (lefthand side in Figure 1), which enables the classifier to take decisions later. The starting point in the training phase is a set of messages, where for each message it is known whether it is valid or not. To be able to use mathematical classification algorithms, numerical dimensions are extracted from the messages in a feature extraction step. The dimension of the feature space is not defined a priori, therefore it creates a vector of features and corresponding values of non-predefined length.

The computational complexity during classification depends on the formulation of the classification model. The complexity is influenced by the number of support vectors needed to describe the model and the number of attributes. Since not all features provide relevant input to the classification model, limiting the amount of features to a minimum set improves performance not only in the training but also in the execution phase. Consequently, in the attribute reduction step, we remove all features (also called attributes) with little or no significance to the decision between valid and invalid messages. Since feature extraction and attribute reduction are the key aspects of our work, we provide more details on them in the following two subsections.

2.2. *Mapping Text Messages into Feature Space*

Feature extraction has to be performed in a way that leads to a good classification result with a reasonable effort and complexity. Starting point for the definition of the features is the idea of using n-grams as proposed by Rieck *et al.* [4]. The authors created a set of n-grams from the training data. The number of different n-grams gives the dimension of the feature space and the occurrence of the n-gram is the associated value. In [4], best performance for SIP messages was achieved with n-gram size $n = 4$, which is also adopted for the work presented here.

Inspired by the idea of using tokens in addition to n-grams [4], we complement the n-grams by special keywords. In contrast to the token idea, our keywords are not determined ad hoc but are predefined based on protocol standards. These keywords are neither limited to a number of occurrences inside the SIP message nor to the associated value. Whenever one of these keywords is recognized during feature extraction, a specific *keyword action* is triggered. Three different keyword actions proved useful:

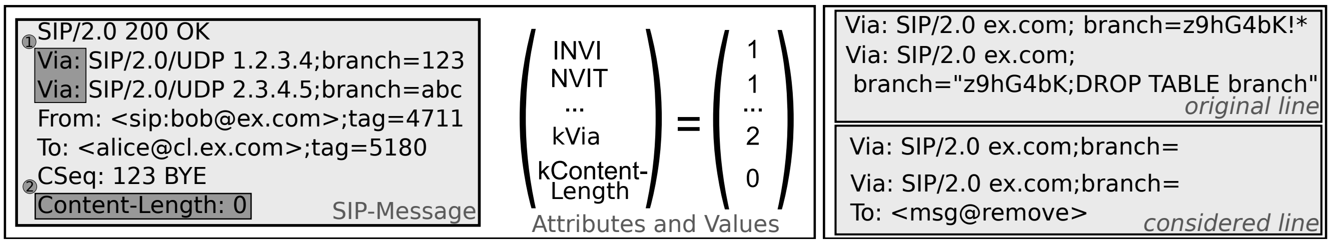
- Keyword action *Counting*. The detection of some protocol-specific tokens contributes more valuable information than the corresponding n-gram combination. For example, a non-standard number of occurrences can be seen as a direct indication of message anomaly. Therefore, we include additional features consisting of protocol-specific keywords and the corresponding number of occurrence in the SIP message. An example is given in Figure 2, where *Via* is chosen as a keyword with keyword action *Counting*. In this example, the value associated with feature *kVia* is two, given that it occurs twice in the message.

- Keyword action *Value extraction*. In some cases, not the number of occurrence of a keyword, but the message contents associated with a keyword are relevant for classification. Therefore, keyword action *Value extraction* includes additional features consisting of the protocol-specific keyword and a value, which is either a keyword-dependent value that directly occurs in the message, or the result of a regular expression over the message contents following this keyword. An example is given in Figure 2 for the *Content-Length* keyword. The feature vector contains the feature *kContent-Length* with value 0, which is the content length value found in the original message.
- Keyword action *Message modification*. Inside the SIP message are parts, which can be interpreted only within the context of the conversation. Given that we do not keep state information about ongoing conversations, we cannot interpret these parts and considering them leads more or less to distracting noise. To avoid producing useless features from these parts of a message, the value can be ignored, replaced with default values, or even be removed completely before n-grams are generated. Before performing such actions, however, local checks of the keyword value are performed to cover anomalies in this part of the message. If this local check fails the message can be directly marked as invalid.

  Figure 2 shows on the right hand side how this keyword action is used with the *branch* keyword. From the SIP standard [1] we take the knowledge that the *branch* value must be globally unique per user agent, has the defined prefix *z9hG4bK*, must only contain characters from a predefined set, and has no length restriction. Thus, when using 4-grams only, we observed that the value of this field adds a high number of n-grams, which are basically useless noise, and can hardly be reduced in the attribute reduction step.

  Therefore, we remove the value which belongs to *branch* after we performed a syntactic check based on the characters allowed. Figure 2 illustrates this keyword action: the *branch* value of the first *Via* line is removed, while the branch value of the second *Via* line does not pass the syntax check. The message containing the latter line will be marked invalid, which is illustrated by replacing the branch value by a corresponding marker in Figure 2.

Figure 2. Examples illustrating the keyword action.



①
SIP/2.0 200 OK
Via: SIP/2.0/UDP 1.2.3.4;branch=123
Via: SIP/2.0/UDP 2.3.4.5;branch=abc
From: <sip:bob@ex.com>;tag=4711
To: <alice@cl.ex.com>;tag=5180
②
CSeq: 123 BYE
Content-Length: 0
SIP-Message
INVI
NVIT
...
kVia
kContent-Length
=
1
1
...
2
0
Attributes and Values
Via: SIP/2.0 ex.com; branch=z9hG4bK!*
Via: SIP/2.0 ex.com;
branch="z9hG4bK;DROP TABLE branch"
original line
Via: SIP/2.0 ex.com;branch=
Via: SIP/2.0 ex.com;branch=
To: <msg@remove>
considered line

The feature extraction algorithm uses a sliding window over the message and detects the current 4-gram or keyword. If the corresponding *keyword action* results in a value, we add it to the feature vector with the keywords name (+prefix) and the determined value (see feature vectors in Figure 1). Based on the need of the SVM for a fixed size feature space, keywords are only added in the training phase. In the execution phase (right part of Figure 1), only these predefined keywords can be used.

2.3. Attribute Reduction

While feature extraction addresses detection performance in terms of good FPR and TPR performance, another target was to maximize throughput. The number of attributes that have to be processed has significant influence on classification time and hence limits throughput. Therefore, we reduce the number of attributes as follows: For every attribute, we compute the attribute weights w_{valid} for valid and w_{invalid} for invalid messages over the whole training data set. The attribute weights are defined as:

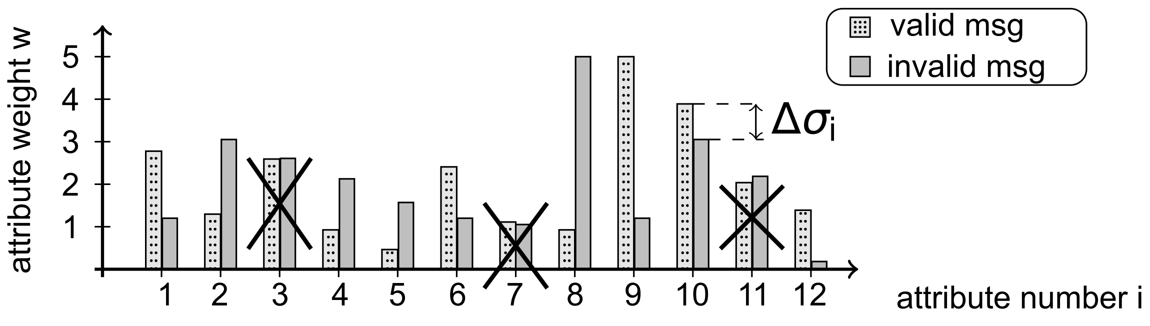
$$w_{\text{valid,i}} = \frac{1}{N_{\text{valid}}} \sum_{j=1}^{N_{\text{valid}}} x_{j,i} \qquad \text{and} \qquad w_{\text{invalid,i}} = \frac{1}{N_{\text{invalid}}} \sum_{j=1}^{N_{\text{invalid}}} x_{j,i}$$

where $x_{j,i}$ denotes the number of occurrences of attribute i in message j. To decide whether an attribute i is significant or not, we calculate the difference $\Delta\sigma_i$ between the attribute weights of valid and invalid messages:

$$\Delta\sigma_i = |w_{\text{valid,i}} - w_{\text{invalid,i}}|$$

If $\Delta\sigma_i$ is below a predefined threshold ϵ, attribute i is removed from the feature vector. Figure 3 illustrates the attribute reduction. In this example, attributes 3, 7 and 11 are not significant for a distinction between valid and invalid messages. The difference $\Delta\sigma_i$ for these attributes is below the threshold ϵ and we thus remove the attributes from the feature vector.

Figure 3. Example of the attribute reduction step to increase throughput.



3. Evaluation

We implemented the feature extraction and attribute reduction in Java. We used the *weka* [5] and *libsvm* [6] libraries for SVM training and evaluation. For training and test, different traces with valid and invalid SIP messages have been used. The used traces were derived together with security experts of a tier one national carrier. Their experts recorded a typical test session which is used to qualify SIP network elements. We used two different training data sets: trace 1 contained 610 messages (12 invalid), and trace 2 contained 928 messages (28 invalid). The true positive rate (TPR) and false positive rate (FPR) in our tests have been obtained by using a test trace containing 12,923 messages (10,000 invalid).

A major problem was setting the SVM parameters in a way to ensure a correct classification for different kind of traces, *i.e.*, when considering different SIP protocol extensions. *Libsvm* trains a model based on the given *l* input data ($\mathbf{x}_i$; $i = 1, ..., l$), each with corresponding class (y_i ; $y_i \in \{1, -1\}$). For the determination *libsvm* solves the following primal problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \qquad \text{subject to} \qquad y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$$

with $\mathbf{w}$ as normal vector of the separating hyperplane, b as displacement of $\mathbf{w}$, ξ_i as slack variable according to [3] and $\phi(x)$ as mapping function into the high dimensional space. This leads to the non-parametric decision function:

$$\operatorname{sgn}\left(\sum_{\substack{support\\vectors}} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) \; + \; b\right) \qquad \text{with } K(a, b) \text{ as the used kernel}$$

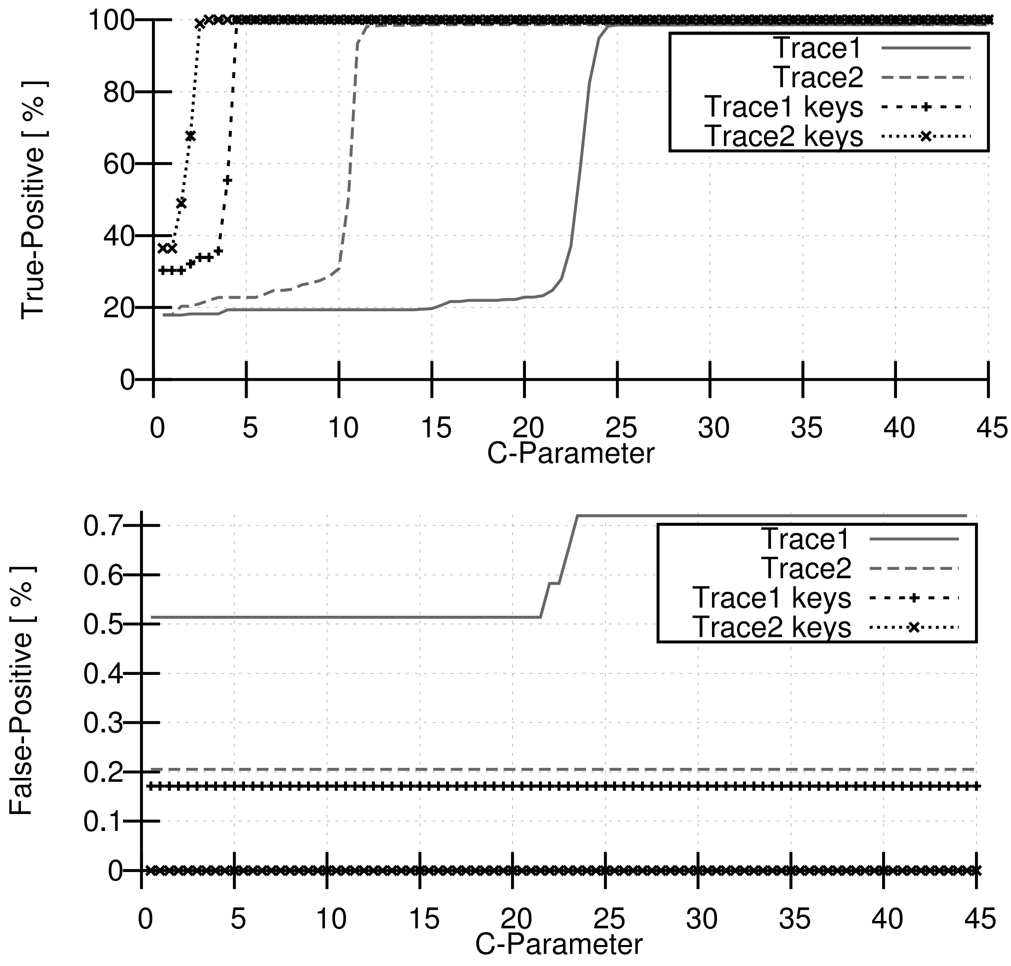
Based on this formulation, we have to choose *C*, the kernel and based on the kernel selection its parameter. We use the default *libsvm*-kernel (RBF-kernel defined as $K(a, b) = \exp(-\gamma\|a - b\|^2)$). The C-parameter describes the ratio between the usage of the hyperplane and additional support vectors [3]. This parameter has shown the largest influence on detection accuracy. Therefore, we left the kernel and γ to the default values ($\gamma = (\#features)^{-1}$) and changed only C during evaluations. *Libsvm* uses C-Parameter settings between 0 and ∞, which corresponds to the interval between 0 and 1 that is mostly used in literature.

For our evaluations, we use an SVM classifier based solely on 4-grams as our reference. We refer to this as the *basic* approach, *i.e.*, without keyword extraction and attribute reduction. Figure 4 shows TPR (top chart) and FPR (bottom chart) for two different traces. The basic approach is indicated with gray lines. The result achieved with our keyword extension is presented with black lines and markers. We can observe an increase of the TPR close to 100% with increasing C-Parameter. There is a steep increase of the TPR for the basic approach at a C-parameter of 10 for trace 2 and at a C-Parameter of 22 for trace 1. The FPR is very low and increases for trace 1 at a C-Parameter of 22 from 0.51% to 0.71%. For trace 2 the FPR stays constant at about 0.2%. We realized that this constant FPR is due to message properties that are covered only partly in the traces. A better coverage of these properties in the training set could improve detection accuracy here. Such a problem arises if e.g. additional optional parameters of a SIP-Header are not contained in the training set. An example is the optional *user* parameter in the following two *From* definitions:

```
From: Bob $<$sip:Bob@ex.com:5060;user=phone>;tag=4711
From: Bob $<$sip:Bob@ex.com:5060>;tag=4711
```

As we can see from Figure 4, setting the C-Parameter for trace 2 is unproblematic, but for trace 1 there is a very small gap between the rising slopes of the TPR and FPR. This means, that the classifier is not stable for this trace. Considering the keyword extension, we see that the FPR is zero for trace 2, *i.e.*, exploiting the message structure helps solving the message property problem mentioned above. The FPR for trace 1 is also considerably reduced and independent of the C-parameter. The charts indicate that our approach of taking keywords into account leads to better detection rates and more stable parameter settings.

Figure 4. Comparison of the C-parameter impact with and without the keyword extension: true positive rate (top), false positive rate (bottom).



We evaluated the performance of our Java implementation on a machine with an Intel Core2Duo E8600 3GHz CPU, 4 GB RAM, a 32Bit Gentoo Linux and SUN JDK 1.6.0.20. Our prototypical implementation shows an average throughput of 2.18 Mbps for the basic approach without attribute reduction and without keywords. With the introduction of the keywords the average throughput decreases to 1.73 Mbps. Using keywords and attribute reduction, we achieved a much higher throughput of 48.73 Mbps. We only applied the attribute reduction step to the n-grams and not to the keywords. We did not observe any influence of the attribute reduction on detection performance (FPR and TPR), hence the choice of the C-Parameter is not affected.

4. Conclusions

We have presented a novel SVM-based anomaly detection algorithm for text-based protocols. In addition to using n-grams, we propose to generate additional features from message properties. A subsequent attribute reduction step then reduces the number of features to a reasonable set. In our evaluations for the SIP protocol, our algorithm achieves lower FPR and higher throughput compared to previously known algorithms.

References

1. Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; Schooler, E. SIP: Session Initiation Protocol, RFC 3261, IETF, June 2002.
2. Nassar, M.; State, R.; Festor, O. Monitoring SIP Traffic Using Support Vector Machines. In *Proceedings of 11th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Cambridge, MA, USA, 15–17 September 2008.
3. Tax, D.M.J.; Duin, R.P.W. Support vector domain description. *Pattern Recogn. Lett.* **1999**, *20*, 1191–1199.
4. Rieck, K.; Wahl, S.; Laskov, P.; Domschitz, P.; Mueller, K.-R. A Self-Learning System for Detection of Anomalous SIP Messages. Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks. In *Proceedings of Second International Conference*, Heidelberg, Germany, 1–2 July 2008; pp. 90–106.
5. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *SIGKDD Explorations* **2009**, *11*, 10–18.
6. Chang, C.-C.; Lin, C.-J. LIBSVM: A library for support vector machines. Available online: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ (accessed on 16 December 2010).