Monitoring SIP traffic using Support Vector Machines

Mohamed Nassar, Radu State, Olivier Festor

Monitoring SIP Traffic Using Support Vector Machines

Mohamed Nassar, Radu State, and Olivier Festor

Centre de Recherche INRIA Nancy - Grand Est
615, rue du jardin botanique, 54602
Villers-Lès-Nancy, France

Abstract. We propose a novel online monitoring approach to distinguish between attacks and normal activity in SIP-based Voice over IP environments. We demonstrate the efficiency of the approach even when only limited data sets are used in learning phase. The solution builds on the monitoring of a set of 38 features in VoIP flows and uses Support Vector Machines for classification. We validate our proposal through large offline experiments performed over a mix of real world traces from a large VoIP provider and attacks locally generated on our own testbed. Results show high accuracy of detecting SPIT and flooding attacks and promising performance for an online deployment are measured.

1 Introduction

The voice over IP world is facing a large set of threats. SPAM on email systems takes a new and very annoying form on IP telephony advertising. This threat is known as SPIT (Spam over Internet Telephony). However, SPIT is not the only threat vector. The numerous software flaws in IP phones and servers affect their reliability and open the door to remotely attack previously unseen in the "stable" world of telecommunication operators (PSTN), which was based on mutual trust among few peers. Leveraging the IP to support voice communications exposes this service (voice) to the known denial of service attacks that can be easily implemented by service or network request flooding on the Internet. Resource exhaustion thus automatically finds its place against SIP proxies and back-to-back user agents, which are essential to support this critical infrastructure. The list of potential threats is huge and ranges from VoIP bots (that could spread by malware and perform distributed attacks, perform SPIT or toll fraud), to eavesdropping and Vishing (similar attack to the Phishing are using VoIP as the transport vehicle) [1].

Securing VoIP infrastructures constitutes one of the major challenges for both the operational and research communities because security by design was not a key component in the early phases of both VoIP research and development. VoIP-specific security solutions are currently required by the market because the research and standardization efforts are still trying hard to address the issues of securing and monitoring VoIP infrastructures.

Our work fits into these efforts and addresses a new monitoring approach for VoIP specific environments. Our monitoring scheme is based on Support Vector Machines for efficient classification. We continuously monitor a set of 38 features in signaling time slices and use these features as the raw input to the classification engine. A threshold based alarm generator is placed on top of the classification engine. We show that the system is both efficient and accurate and study the impact of the various features on the efficiency.

We start the presentation with a short survey on VoIP security with focus on flooding attacks and SPIT. We then give a functional description of our monitoring solution together with the definition of the 38 features computed in our system for classification (section 3). In section 4, we provide a short mathematical background of the SVM learning machine model used in the monitoring process. Offline traces inspection is presented in section 5 where we also describe the data set. Section 6 demonstrates the performances of our approach to detect different types of attacks. Related work is addressed in section 7. Section 8 concludes the paper and enumerates some future work.

2 The Threat Model

2.1 Flooding Attacks

Denial of service attacks can target the signaling plane elements (e.g. proxy, gateway, etc.) with the objective to take them down and produce havoc in the VoIP network. Such attacks are launched by either flooding the signaling plane with a large quantity of messages, malformed messages or executing exploits against device specific vulnerabilities.

The authors of [2] categorize some of these attacks based on the request URI and perform a comparative study of these ones against popular open source VoIP equipment. We adopt the same categorization, i.e.:

- UDP flooding: Since the vast majority of SIP systems use UDP as the transport protocol, a large amount of random UDP packets are sent in an attempt to congest the network bandwidth. Such attacks produce a high packet loss. Legitimate call signaling has thus a reduced probability to reach the target and to be processed.
- INVITE flooding with a valid SIP URI: The attacker calls one user/phone registered at a server/proxy. The proxy relays the calls to the phone. If the proxy is stateful it will manage a state machine for every transaction. The phone is quickly overloaded by the high rate of calls and is no more able to terminate the calls. As a result, the server is allocating resources for a long time and it will run out of memory.
- INVITE flooding with a non existent SIP URI: If the attacker doesn't know a valid SIP URI registered on the target, it can send calls to an invalid address. The proxy/server responds with an error response like "user not found". When the attack rate is higher than the server capabilities, the resources are exhausted. This type of flooding is less disturbing than the previous one but

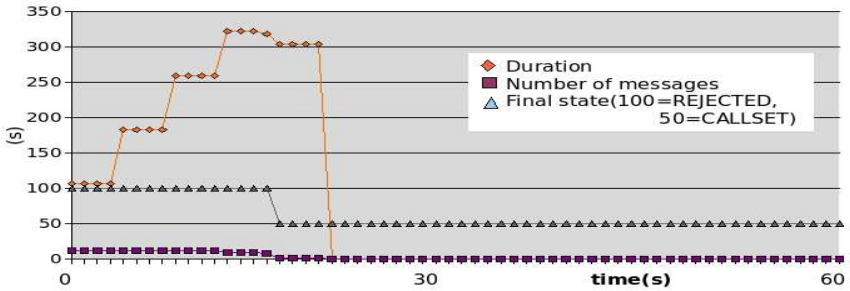
the target CPU is loaded with useless transactions and legitimate requests may be rejected.

- INVITE flooding with an invalid IP domain address: The attacker calls a user with a rogue IP address of the destination domain. The target is led to connect several times to an unreachable host/network while keeping the state of the current SIP transaction. This attack is efficient on some proxies like OpenSER [2].
- INVITE flooding with an invalid domain name: The attacker calls a user with a false destination domain name. The target is trapped to send DNS requests to resolve the domain name. The target may issue different DNS types (A, AAAA, SRV, NAPTR, ENUM) and repeat them multiple times. In the same time, the target is managing the transactions waiting for a valid DNS response to proceed. Memory is quickly exhausted. The effect of this attack on the performance of OpenSER is shown in Fig. 1. The impact is evaluated in terms of duration, number of messages exchanged and final state of sessions or transactions. The behavior of the server can be divided in two successive phases. In the first phase, the first few requests are correctly handled (REJECTED) but the session duration is increasing and the proxy is slowing down. The number of messages is increasing because of response retransmissions (no ACK is sent by the attacker). In the second phase, the proxy is no more able to handle the requests (still in CALLSET state) so the proxy is taken down. The take down time is about 20 seconds for an attack having just one INVITE/s rate.
- INVITE flooding with an invalid SIP URI in another domain: The attacker calls a user/phone located in another domain than the target's one. The target relays all requests to the server/proxy of the other domain. The latter replies with an error response. In this way, multiple targets are hit at the same time and cascading failures occur.
- INVITE flooding with a valid SIP URI in another domain: The attacker calls a user/phone registered in another domain. The target relays all requests to the server/proxy of the other domain which sends them to the phone. The phone gets quickly out of service and maintaining the state by the intermediary servers will exhaust the resources from all the servers in the forwarding chain.
- INVITE/REGISTER flooding when authentication is enabled: The attacker sends INVITE or REGISTER messages and then stops the handshaking process. The proxy/registrar responds with a challenge and waits for the request to be send again with the proper authentication credentials. This process is costly for the proxy/registrar in term of computing (generating challenges and nonces) and memory (dialogs/transaction state machines).

2.2 Social Threats and SPIT

Social threats are attacks ranging from the generation of unsolicited communications which are annoying and disturbing for the users to more dangerous

Fig. 1. OpenSER Response to an INVITE Flooding with Invalid Domain Name



data stealing (Vishing) attacks. The threat is classified as social since the term "unsolicited" depends on user-specific preferences. This makes this kind of attack difficult to identify. An example of this is a threat commonly referred to as SPam over Internet Telephony (SPIT). This threat is similar to spam in the email systems but is delivered by means of voice calls. This leverages the cheap cost of VoIP when compared with legacy phone systems. It's currently estimated that generating VoIP calls is three order of magnitude cheaper than generating PSTN calls. Such SPIT calls can be telemarketing calls that sell products. A subtle variant of SPIT is the so-called Vishing (VoIP phishing) attack, which aims either to make the callees dial expensive numbers in order to get the promised prize or to collect personal data redirecting the users towards an Interactive Voice Responder (IVR) pretended to be trusted. Most of these attacks are going to be generated by machines (bot-nets) programmed to do such a job. Unsolicited communications (like SPIT or Vishing) are, from a signalling point of view, technically correct transactions. It is not possible to determine from the INVITE message (in the case of SIP) if a VoIP transaction is SPIT or not. From a technical point of view, the challenge is actually higher since the content is not available to help in the detection until the phone rings (disturbing the user) and the callee answers the call. For this reason, techniques successfully used against e-mail spam like text filtering are hardly applicable in the VoIP sphere. Even if a transaction is identified as unsolicited how to handle such a transaction highly depends on the legal environment in the country of the caller.

3 Our Monitoring Solution

When facing the mentioned threats, monitoring of the signalling traffic can detect anomalous situations and prevent them. The monitoring scheme can be quite simple and flexible to support different techniques. Thus, our approach follows these principles. As shown in Fig. 2, we track SIP messages in a queue of predefined size. Once the queue is full, this slice of messages is used to compute a vector of statistics/features. The classifier decides if a vector represents a certain anomaly and issues an alarm event if necessary. This approach is based on a learning phase in which couples (vector, class Id) have been used to feed

the engine for learning. This learning process can be made on the fly during the operational phase of the monitoring system by allowing it to update the prediction model over time. Finally, an event correlator or decider has to filter and correlate the events. It generates an alarm for a group of events if they trigger one of the rules/conditions. e.g. if the number of events of type i bypasses a certain threshold in a period of time t.

The architecture is modular and enables experimenting with different classification and artificial intelligence techniques ranging from statistics and information theory to pattern classification and machine learning. The pace of the system t_{pace} is the time it takes to make a decision about one slice without accounting for the time needed by the event correlation stage. This time is composed of two components: the analysis time of the processor and the machine time of the classifier. The design achieves real time pace if t_{pace} is less than the size of the slice S divided by the arrival rate of messages λ:

$$t_{pace} = t_{analysis} + t_{machine}$$
$$t_{pace} < \frac{S}{\lambda}$$

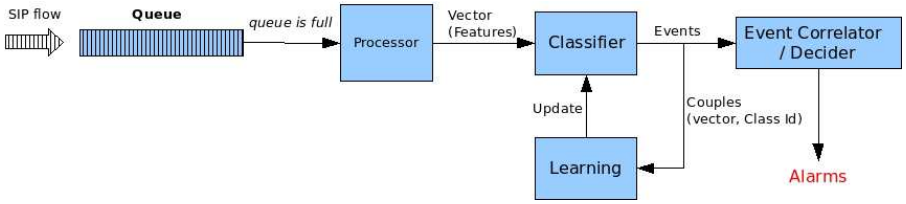
We define in the following the important features that characterize a slice of SIP traffic and motivate why we collect them. We divide these features in four groups:

- **General statistics :** are number of requests, number of responses, number of requests carrying an SDP (Session Description Protocol) body, average inter requests arrival time, average inter response arrival time and average inter requests arrival time for requests having SDP bodies; these statistics represent the general shape of the traffic and indicate the degree of congestion. The fraction of requests carrying SDP bodies (normally INVITE, ACK or UPDATE) is a good indicator because it will not exceed a certain threshold. An excessive use of re-INVITE or UPDATE for media negotiation or maybe QoS theft increases the number of SDP bodies exchanged and decrements the average inter-arrival of them. Flooding attacks are associated with peaks of all these statistics.
- **Call-Id based Statistics:** are number of Call-Ids, average of the duration between the first and the last message having the same Call-Id, the average number of messages having the same Call-Id, the number of different senders (the URI in the From header of a message carrying a new Call-Id) and the number of different receivers (the URI in the To header of a message carrying a new Call-Id). Similar to the Erlang model used in the telecommunication networks, where the arrival rate of calls and the average duration of a call characterize the underling traffic, the arrival rate of Call-Ids (can be starting a call or any kind of SIP dialog) and the interval time of messages having the same Call-Ids, can be used to characterize the overlay SIP traffic. Nevertheless, we notice that non-INVITE dialogs have shorter durations and fewer

number of messages than INVITE dialogs. Thus their Call-Id statistics can be taken as different features.

- **Distribution of final state of dialogs/Call-Ids:** Since we are using a limited number of messages in the traffic analysis unit, dialogs can be partitioned into two or several units/slices. The final state of a dialog at the analysis moment is considered and this one is not necessarily the final state when all the messages of the dialog can be taken into account. The following states are defined: NOTACALL: for all non-INVITE dialogs, CALLSET: for all calls/INVITE dialogs that do not complete the initiation, CANCELED: when the call is cancelled before it is established, REJECTED: for all redirected or erroneous sessions, INCALL: when the call is established but not realized yet, COMPLETED: for a successful and ended call and RESIDUE: when the dialog does not start with a request. This latter is a residual of messages in a previous slice. In a normal situation where the size of the unit is large enough, NOTACALL, COMPLETED and REJECTED (in busy or not found situations) dominate this distribution. Major deviations may indicate an erroneous situation.
- **Distribution of SIP requests:** are INVITE, REGISTER, BYE, ACK, OPTIONS, CANCEL, UPDATE, REFER, SUBSCRIBE, NOTIFY, MESSAGE, INFO, PRACK. Although the first five types represent the main methods used in SIP, every other type may point out a specified application running above. The number of REGISTER sent by a user within a time interval is indirect proportional to the period of registration (`expires` parameter or `Expires` header). Obviously, the total number of REGISTER messages is proportional to the number of users of the domain and inversely proportional to the average period of registration among all users. The existence of SUBSCRIBE and NOTIFY messages indicates SIP presence services. Instant messaging can also be revealed by MESSAGE requests. REFER requests may reveal a SIP peer to peer application or some call transfer running above. INFO requests are normally used to carry out of band DTMF tones within PSTN-VoIP calls. Finally, PRACK requests may reveal VoIP to PSTN activity.
- **Distribution of SIP responses:** are Informational, Success, Redirection, Client Error, Server Error, Global Error. An unexpected high rate of error responses is a good indication for error situations.

Fig. 2. Real-time Online SIP Traffic Monitoring



Among the different scientific approaches in the area of classification (Bayesian networks, decision trees, neural networks), we have chosen the support vector machines approach for their superior ability to process high dimensional data [3, 4]. SVM is a relatively novel (1995) technique for data classification and exploration. It has demonstrated good performance in many domains like bioinformatics and pattern recognition (e.g. [5] and [6]). SVM has been used in network-based anomaly detection and has demonstrated better performance than neural networks in term of accuracy and processing proficiency [7]. In the next section, we give a short description of the SVM concept and methodology.

4 Support Vector Machines

Principle Given a set of couples $S = (\overrightarrow{x_l}, y_l)_{1 \leq l \leq p}$, with $y_l \in \{-1, +1\}$, which denotes the correct classification of the training data, the SVM method tries to distinguish between the two classes by mean of a dividing hyperplane which has as equation $\overrightarrow{w}.\overrightarrow{x} + b = 0$. If the training data are linearly separable, the solution consists in maximizing the margin between the two hyperplanes, $\overrightarrow{w}.\overrightarrow{x} + b = +1$ and $\overrightarrow{w}.\overrightarrow{x} + b = -1$, such that for all points either $\overrightarrow{w}.\overrightarrow{x} + b \geq +1$ (1) or $\overrightarrow{w}.\overrightarrow{x} + b \leq -1$ (2). This is equivalent to minimizing the module $|\overrightarrow{w}|$ because the distance between the two mentioned hyperplanes is $2/|\overrightarrow{w}|$. The resulting quadratic problem where the conditions (1) and (2) are aggregated is formulated as:

$$\boxed{\begin{array}{c}\text{Find } \overrightarrow{w} \text{ and } b \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} \\ \text{so that } y_l(\overrightarrow{w}.\overrightarrow{x_l}) + b \geq 1 \forall (\overrightarrow{x_l}, y_l) \in S\end{array}}$$

The non linear separation has a similar formulation except that we replace the dot product by a non-linear kernel function. The kernel function takes the data set to a transformed feature space where it searches the optimal classifier. The transformation may be non-linear and the transformed space high dimensional. The maximum-margin hyperplane in the high-dimensional feature space may be non-linear in the original input space. The following kernels can be used :

- linear $K_l(\overrightarrow{x}, \overrightarrow{z}) = \overrightarrow{x}.\overrightarrow{z}$
- polynomial $K_d(\overrightarrow{x}, \overrightarrow{z}) = (\gamma \overrightarrow{x}.\overrightarrow{z} + r)^d$, $\gamma > 0$
- radial basis function $k_{rbf}(\overrightarrow{x}, \overrightarrow{z}) = exp(-\gamma|\overrightarrow{x} - \overrightarrow{z}|^2)$ where $\gamma > 0$
- sigmoid $k_s(\overrightarrow{x}, \overrightarrow{z}) = tanh(\gamma \overrightarrow{x}.\overrightarrow{z} + r)$, $\gamma > 0$ and $r < 0$

The C-SVC (C parameter - Support Vector Classification) approach is particularly interesting when the training data is not linearly separable.

C-SVC For the general case where the data S is not separable, the solution allows some points to be mislabeled by the separating hyperplane. This method, so called soft margin, is expressed by the introduction of slack variables ξ_l where ξ_l measures the degree of misclassification of a point x_l. The objective function is then increased by a function which penalizes non-zero ξ_l, and the optimization becomes a trade off between a large margin, and a small error penalty.

$$\boxed{\begin{array}{l}\text{Find } \overrightarrow{w}\text{, b and } \xi \text{ to minimize } \frac{1}{2}\overrightarrow{w}.\overrightarrow{w} + C\sum_l \xi_l \\ \text{so that } \begin{cases} y_l(\overrightarrow{w}.\overrightarrow{x_l}) + b \geq 1 - \xi_l, \forall(\overrightarrow{x_l}, y_l) \in S \\ \xi_l \geq 0, \forall l \end{cases}\end{array}}$$

5 Monitoring SIP

We aim to detect anomalies within a SIP traffic capture, demonstrate the accuracy of the learning machine to identify attacks and non-attacks and distinguish between different types of attacks. We have performed an extensive analysis on offline VoIP traces in order to assess the performance of our approach.

We use the popular LibSVM tool [8] which contains several algorithms for classification and regression. In addition, the tool provides support for multi-class classification and probability estimates (so a test vector x_i seems to be of class i with a probability p_i) as well as support for one class SVM training. LibSVM is bound to other several tools such as an algorithm that performs a grid search over the SVM parameters space and optimizes their values by cross validation (divide the data into n subsets, for i going from 1 until n, learn over all the subsets except subset number i then test over subset number i). At the end, we can measure the test accuracy for each subset. The aggregation of all results is the accuracy given by the selected parameters. In Fig. 3 we illustrate this tool's flow. The data we use in this study originates from two different sources. The

Fig. 3. SVM Flow Chart

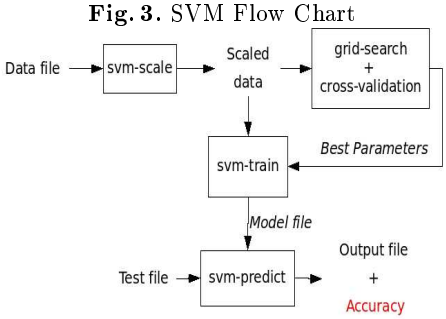
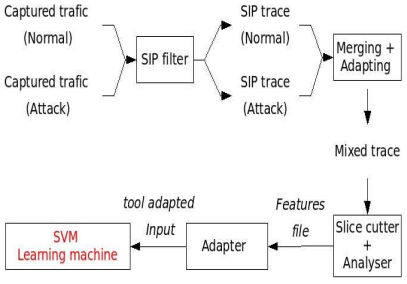


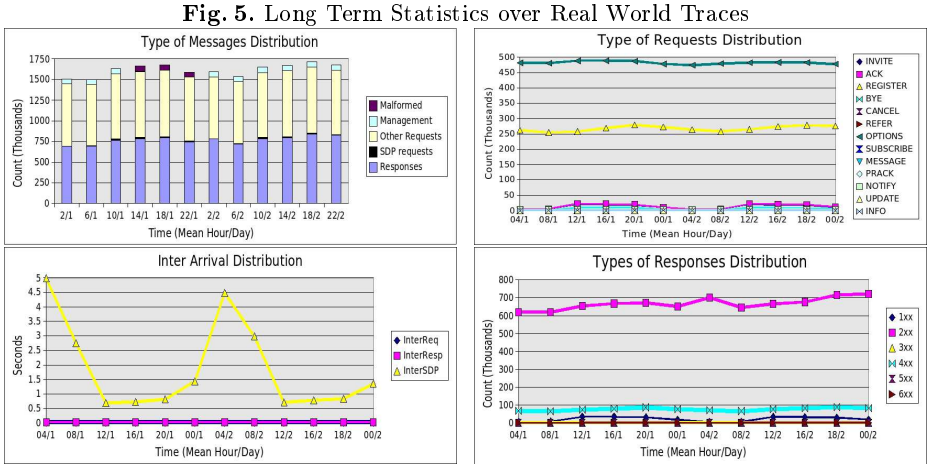
Fig. 4. Analysis Flow Chart



first source is traffic from a real-world VoIP provider and it is supposed to be completely normal. The second source is signaling traffic from a small test-bed installed by us to generate different forms of SIP attacks. We have three types of data files: clean and normal trace, clean attack trace, and mixed trace which is a normal trace where attack is injected.

To be processed by the SVM tool, each data file is cut into slices and entered into the analyzer. For each slice, the analyzer evaluates a set of predefined features (38 variables are defined in our study) and builds a vector for the LibSVM. All vectors are assembled in one file and annotated as either attack vector or normal vector. In Fig. 4, this process is shown for a mixed trace.

Fig. 5. Long Term Statistics over Real World Traces



5.1 Normal Traffic

The input data is a SIP trace from a two days continuous capture at a real world VoIP provider server. We performed a preliminary long term analysis of the traces with a two hours step. We depict the results in the four charts of Fig. 5. If we consider the distribution of different SIP messages, we can remark the following:

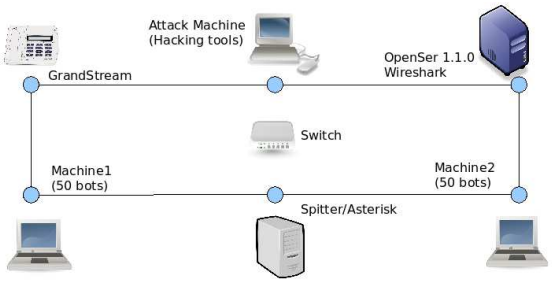
- The two main components of the traffic are the OPTIONS messages in the first place and then the REGISTER messages.
- Some methods are absent from the capture such a MESSAGE, PRACK and UPDATE.
- Some methods like NOTIFY have constant statistics over all periods of the day which reveal SIP devices remaining always connected and periodically sending notifications.
- The three main components of the call signalling (INVITE, BYE and ACK) have practically constant ratios over all the slots, with an average ratio $\#INVITE/\#BYE = 2.15$ and $\#INVITE/\#ACK = 0.92$.

Response distribution is dominated by the 2nd response class (most of them belong to OPTIONS and REGISTER transactions). 3xx, 5xx and 6xx are very rare while informational responses (1xx) follow INVITE messages because they are exclusively used in INVITE transactions (the average ratio $\#INVITE/\#1xx = 0.59$ can be explained by the fact that a call probably regroups one 100 Trying and one 180 Ringing so two 1xx responses). Average inter-request arrival and average inter-response arrival seem to be constant over all periods and they are about 20 ms. While average inter-request carrying SDP bodies which are exchanged in call dialogs move inversely to the quadruple (INVITE-BYE-ACK-1xx) curve, they reach 3s in quiet hours and decrease to 0.5s in rush hours.

5.2 The Testbed

The testbed consists of one OpenSER server and three other machines: the first machine plays the role of the attacker and uses a number of hacking tools (scanning, flooding, SPIT). The two other machines play the role of victims where one hundred SIP bots are equally distributed and running. The bots are programmable SIP agents and are controlled by an IRC channel[1]. All SIP bots and a GrandStream hardphone are registered to the OpenSER server and all machines belong to the same domain. Traces of attacks performed by the attacker machine are collected at the OpenSER server.

Fig. 6. Testbed of Attack Generation



6 Performance and Accuracy

All experiments are done in a machine which has an Intel Pentium 4 CPU 3.40GHz and 2GB RAM memory running a Linux kernel 2.6.18-1. In term of performance, experiments show that a file containing 2449 slices/vectors of 38 features takes between 196 and 994 ms in the SVM prediction stage (depending on the used kernel).

Coherence Test

The first question we addressed was how many of the normal traces are self-similar and consistent. For example, is traffic from 22:00 to 02:00 from a day similar to traffic of the same period in another day? To test the coherence between two given traces, we used the following procedure: the analyzer evaluates feature vectors from each trace. Vectors are then labeled with respect to the origin trace and scaled. We make a 2-fold training test over all the vectors. In a 2-fold test, training is done over one part of the file and the testing is performed over the second. We define the coherence to be indirect proportional to

[1] `http://www.loria.fr/~nassar/readme.html`

the resulting accuracy of the 2-fold cross training. As long as the SVM can not distinguish between the two traces, they are tagged to the same class. In Table 1, we summarize some results: We tested the coherence of a period with respect to

Table 1. Coherence Test for two Successive Days

Day 1	06-10	10-14	14-18	18-22
Day 2	06-10	10-14	14-18	18-22
Accuracy(%)	55.91	53.72	52.83	56.90

other periods. In Table 2, we show the results of the same procedure for a period of 2-6 of Day 1 compared to other periods of the same day. SVM is not able to

Table 2. Coherence Test for Different Periods of the Same Day

Day 1	02-06	02-06	02-06	02-06	22-02
Day 1	06-10	10-14	14-18	18-22	22-02
Accuracy(%)	51.82	62.79	63.72	63.76	60.80

label 50% of vectors in the correct class while proceeding with the same period of two successive days and 40% of vectors during different periods of the same day. The second table reveals that period 02-06 is more coherent with neighboring periods (06-10 and 22-02) than with other periods of the day. In conclusion, the coherence of the data is acceptable.

Multi-Class Detection Experiment

We also tested SVM's ability to distinguish between different classes of traffic: for instance traces coming form different VoIP platforms. We built a group of four traces, each representing a different traffic class : normal traffic, a burst of flooding DoS, a trace generated by the KIF stateful fuzzer [9], and a trace generated by an unknown testbed as shown in Table 3. The size of the analyzed slice is fixed to 30 messages. After analysis, a 2-fold training/testing cross test is performed over all respectively labeled vectors (2449 vectors). The test Accuracy is defined as the percentage of correctly classified vectors over all test vectors. When the RBF (Radial Basis Function) kernel is used with default parameters (C=1 and $\gamma = 1/38$), the accuracy is 98.24%.

Comparison between Different Kernel Experiments

The RBF kernel is a reasonable first choice if one can assume that the classes are not linearly separable and because it has few numerical settings (small number of

Table 3. Multi-Class SIP Traffic Data Set

Trace	Normal	DoS	KIF	Unknown
SIP pkts	57960	6076	2305	7033
Duration	8.6(min)	3.1(min)	50.9 (min)	83.7(day)

parameters, exponential function bounded between 0 and 1). On the other hand, linear and RBF kernels have comparable performance if the number of features is significantly higher than the number of instances or if both are to large [8]. Therefore, we have tested all kernels with their default parameters over our dataset. The accuracy (defined as the percentage of correctly classified messages over all the test results) for 2-fold cross and machine dependent running time are shown in Table 4. The last two lines of the table are for RBF and linear kernels

Table 4. Testing Results for Different Kernels

Kernel	Parameters	Accuracy(%)	Time(ms)
Linear	$C = 1$	99.79	196
Polynomial	$C = 1$; $\gamma = 1/38$; $r = 0; d = 3$	79.09	570
Sigmoid	$C = 1$; $\gamma = 1/38$; $r = 0$	93.83	994
RBF	$C = 1$; $\gamma = 1/38$	98.24	668
Linear	$C = 2$	99.83	157
RBF	$C = 2$; $\gamma = 0.5$	99.83	294

after parameter selection. Machine running time is given for comparison purpose only and it is averaged over ten runs. RBF and linear kernels have clearly better accuracy and execution time. We expect that RBF kernel will bypass linear kernel performance when dealing with larger sets of data.

Size of SIP Slice Experiment

The analyzer window is an important parameter in the feature evaluation process. The size of the slice can be fixed or variable with respect to other monitoring parameters. In this experiment, we report the accuracy of our solution, when changing the size of the analyzed slice. The results shown in Table 5 were obtained using a 5-fold cross test using a RBF kernel and the default parameters. The time the analyzer takes to process the packets is critical in online monitoring. This is the reason why we show the analysis time of the overall trace: (note

that values are for comparison purpose). As expected, the accuracy improves with larger window size, which incurs an increased analysis time.

Table 5. Testing Results for Different Kernels

Window size	5	15	30	60	90	120	150
Accuracy (%)	95.4	99.32	99.30	99.67	99.63	100	100
Analysis Time (min)	1.12	2.40	2.56	4.31	6.39	7.42	8.51

Feature Selection

The 38 features are chosen based on domain specific knowledge and experience, but other features might be also relevant. The selection of highly relevant features is essential in our approach. In the following experiments, we rank these features with respect to their relevance. We can thus reduce the number of features by gradually excluding less important features from the analysis. In Table 6, the results of a preliminary experiment, where we exclude one group of features at each column in the following order: the distribution of final state of dialogs, the distribution of SIP requests, the distribution of SIP responses, and the Call-Id based statistics are given. The last column of the table represents only the general statistics group of features. Experiments use a 5-fold cross test over our data set with RBF kernel and its default parameters. The test accuracy is the percentage of correctly classified vectors over all the vectors in the test data set. Although we notice a sudden jump between 12 and 7 features, the associated

Table 6. Results for Decreasing Size of Features Set

# of features	38	31	18	12	7
Accuracy (%)	99.30	99.39	98.90	98.65	98.22
Machine Time (s)	1.85	1.59	1.42	1.28	0.57

accuracy is not strictly decreasing as a function of number of features used. It is thus reasonable to inquire on the dependencies among the features.

Detection of Flooding Attacks

We have used the Inviteflood tool [2] to launch SIP flooding attacks. We have used INVITE flooding with an invalid domain name (which is the most impacting on the OpenSER server). We have generated five attacks at five different rates, where each attack lasts for one minute. After adaptation (we assume that one machine of the real world platform is performing the attack), each one minute

attack period is injected into a normal trace of two hours duration. The time of the attack is fixed to five minutes after the start of the two hours period. Each mixed trace is then analyzed and labeled properly (positively along the period of attack and negatively in all the remaining time).

We have trained the system with the mixed trace (flooding at 100 INVITE/s - normal trace) in the learning stage. This means that 100 INVITE messages are taken as a critical rate (the rate we consider as threshold to launch an alarm).

As shown in Fig. 7 (for simplification and clarity sake a slice is sized to only three packets), we take the period of attack and we calculate the corresponding SVM estimation. The estimated probability is the average of the estimated probabilities for the elementary slices composing the attack traffic. This granular probability is given by the LibSVM tool and is useful for both the probability estimate option in both learning and testing stages. We define the detection accuracy as the percentage of vectors correctly classified as attack over all vectors of the attack period. The results are in Table 7: the detection accuracy-1 is obtained without a parameter selection (Default parameters : $C = 1$, $\gamma = 1/38$, training accuracy: 90.95), detection accuracy-2 and calculated probabilities are after parameter selection ($C = 32$, $\gamma = 0.5$, training accuracy is of 93.93).

Fig. 7. Attack Detection in a Mixed Trace

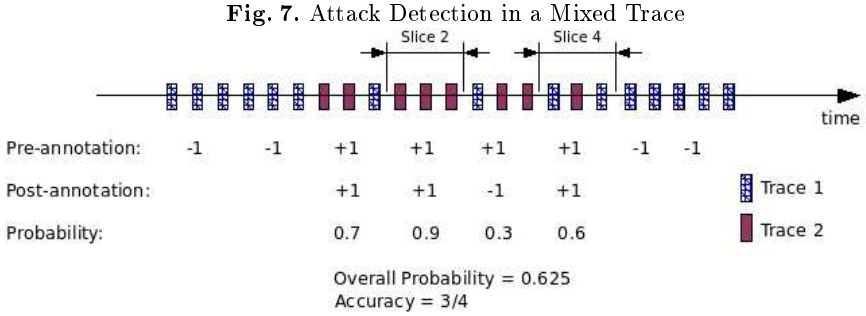


Table 7. Attack Estimation for Different Rates of Flooding

Flooding Rate (INVITE/s)	0.5	1	10	100	1000
Detection Accuracy-1 (%)	0	0	5.47	67.57	97.36
Detection Accuracy-2 (%)	0	1.48	30.13	88.82	98.24
Pr(Normal)	0.96	0.95	0.73	0.24	0.07
Pr(Attack)	0.04	0.05	0.27	0.76	0.93

Even though stealthy attacks cannot to be detected, the results show a promising opportunity to fine-tune the defensive solution. The threshold rate

can be learnt by a dual trace : the ongoing normal/daily traffic and a stress condition where the server was troubleshooted or was noticed to be under-operating. In this way, SVM is promising for an adaptive online monitoring solution against flooding attacks.

Detection of SPIT Attacks

SPIT mitigation is one of the open issues in VoIP security today. Detection of SPIT alone is not sufficient if it is not accompanied by a prevention system. In-depth search in the suspicious traffic is needed to build a prevention system to block the attack in the future. Elements like IP source and URI in the SIP headers can be automatically extracted.

To generate SPIT calls, we used a well known tool which is the Spitter/Asterisk tool [2]. Spitter is able to generate call instances described in a ".call" file using the open source Asterisk PBX. The rate of simultaneous concurrent calls can also be specified as an option of the attack. We profiled our programmable bots to receive SPIT calls. Once an INVITE is received, the bot chooses randomly between three different responses :

- the first choice is to ring for a random time interval between one and six seconds and then to pick up the phone. This emulates two cases : a voice mail which is dumping a message or a human which is responding. The bot then listens during a random time between five and ten seconds and hangs up,
- the second choice is to respond with 'Busy',
- the last choice is to ring for some time and then to send a redirection response informing the caller that the call has to be directed to another destination (destination that we assume to not be served by this proxy). Other similar scenarios like forking (by the proxy) or transferring (by the bot) can also be supported.

We have performed two experiments with two different hit rates. The former is a partial SPIT: Spitter targets the proxy with hundred destinations and among these only ten are actually registered bots. In this case the hit rate is just 10%. This emulates the real world scenario where attackers are blindly trying a list of extensions. The latter is a total SPIT: we assume that attackers knew already the good extensions so the hit rate is 100%. This emulates the real world scenario where attackers knew already the good extensions either by a previous enumerating attack or from a web crawler.

In the partial SPIT experiment (SPIT not covering all the domain extensions, hit rate < 100 %), we send four successive campaigns with respectively one, ten, fifty and hundred concurrent calls. In the first campaign, Spitter does not start a dialog before the previous dialog is finished. In the second campaign, ten dialogs go on at the same time and only when a dialog is finished, a new dialog is started.

The four resulting traces (duration about two minutes each) are injected - after adaptation (we assume that one agent of the real trace is performing the

attack against the hundred other agents) - in four different normal traces (duration of two hours each). The traces are then cut into slices of thirty messages and analyzed. These are annotated positively for the period of attack and negatively in all the remaining duration. The mixed trace with fifty concurrent calls SPIT is used in the training stage. The SVM prediction results are shown in Table 8. True positives are the percentage of vectors correctly classified as attack over all the vectors of the attack period. True negatives are the percentage of vectors correctly classified as normal over all the vectors of the normal period. These

Table 8. Detection of Partial SPIT in Four Mixed Traces With Different Intensities

# of Concurrent Calls	**True Positives (%)**	**True Negatives (%)**
RBF; C= 1; $\gamma = 1/38$; Training accuracy = 99.0249		
1	0 (0/3697)	100
10	1.30 (10/766)	
50	10.01 (62/619)	
100	18.31 (102/557)	
Linear ; C=1 ; Training accuracy = 99.0197		
1	0 (0/3697)	100
10	2.09 (16/766)	
50	10.66 (66/619)	
100	19.39 (108/557)	

results should be considered under the larger umbrella of event correlation. For instance, the example with ten concurrent calls:

- Most of the two hours traffic is normal and is correctly detected (47436 slices).
- 16 out of the 766 slices that compose the attack traffic are detected. This means that we have ten correct events in a period of two minutes, because the detection of one slice is highly relevant to all ongoing traffic around this slice.

In addition, the attacks are partial since they target a small fraction of the users of the VoIP server (more than 3000 users are identified in the two hours period). We agree that a stealthy SPIT of the magnitude of one concurrent call is never detected, but in the case of hundred concurrent calls, one over five positives is successfully detected when training was done using a half of this intensity attack.

With the help of a set of deterministic event correlation rules, our online monitoring system is able to detect the attacks efficiently:

Predicate	**SPIT intensity**
10 distributed positives in a 2 minutes period	Low
Multiple Series of 5 Successive Positives	Medium
Multiple Series of 10 Successive Positives	High

In the full SPIT experiment, we request the hundred bots to register with the proxy. Spitter hits all the bots in four successive campaigns with increasing intensity. Results are slightly better than in the partial SPIT experiment (Table 9). Partial SPIT generates an abnormal traffic at the same level as full SPIT does.

Table 9. Detection of Full SPIT in Four Mixed Traces With Different Intensities

# of Concurrent calls	1	10	50	100
RBF; C= 1; $\gamma = 1/8$; Training accuracy = 98.9057				
True Positives	0.03 2/7015	3.05 15/492	12.18 85/698	23.41 184/786
True Negatives	100			

7 Related Works

VoIP security is a recent research domain that emerged over the last few years with the increasing use of this technology by enterprises and individuals. Combating SPIT and DoS is the subject of many research proceedings. Quittek et al. [10] apply hidden Turing tests and compare the resulting patterns with typical human communication patterns. Passing these tests causes significant resource consumption in the SPIT generation side. The authors of [11] propose a call rank mechanism based on call duration, social networks and global reputation to filter SPIT calls. Other ideas include a progressive and multi (short term -long term) grey level algorithm [12] and incorporating active stack fingerprinting [13].

The authors of [14] design application and transport sensors to protect enterprise networks from VoIP DoS attacks based on previous works on TCP DoS protection and study different recovery algorithms. The authors of [15] modify the original state machine of SIP transactions to detect transaction anomalies and apply different thresholds to detect flooding attacks. More adaptive to such attacks is the work of Sengar et al. [16] where the Hellinger distance between learning and testing periods is used to detect TCP SYN, SIP INVITE and RTP floods. Their approach shows good performances. There have many papers in the community on generic intrusion detection methods [17–19] without to extend to the fine tuned session, dialog, transaction related parameters found in SIP. Over the past, many security related applications have leveraged machine learning techniques and the reader is referred to [20] and [21] for an overview.

The closest work to ours is the study of [22] where the authors have presented a traffic behavior profiling methodology and demonstrated its applications in problem diagnosis and anomaly detection. Our work is more oriented towards attack detection and classification rather than proposing a global and multi level profiling methodology. We have addressed the VoIP specific event correlation and

honeypots in previous published work [23] and [24], which did not cover SIP-level monitoring.

8 Conclusion and Future Works

As attacks on VoIP are popping-up in different forms with increasing impact on both the users and infrastructure, more monitoring and security management is needed. In this paper, we proposed an online monitoring methodology based on support vector machines. Our idea is to cut the ongoing signalling (SIP) traffic into small slices and to extract a vector of defined features characterizing each slice. Vectors are then pushed into a SVM for classification based on a learning model. We then use a deterministic event correlator to raise an alarm when suspicious and abnormal situations occur.

We validated our approach by offline tests over a set of real world traces and attacks which are generated in our customized testbed and inserted in the normal traffic traces. Results showed a real time performance and a high accuracy of detecting flooding and SPIT attacks especially when coupled with efficient event correlation rules. Detection of other types of attacks are future work.

Unsupervised learning techniques are appealing because they don't need a priori knowledge of the traffic and can detect new and previously unknown attacks. We consider currently to redefine and reorder our set of features based on different features selection algorithms. We will extend the current event correlation and filtering algorithm in order to reveal attack strategies and improve intrusion prevention/detection accuracy.

Acknowledgment. We would like to thank Mr Dorgham Sisalem and Mr. Sven Ehlert, both from Fraunhofer Institute in Berlin for their comments and feedback on discussing the analysis of SIP traces.

References

1. VoIPSA: VoIP security and privacy threat taxonomy. Public Realease 1.0 (Oct 2005) `http://www.voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf`.
2. Endler, D., Collier, M.: Hacking Exposed VoIP: Voice Over IP Security Secrets and Solutions. McGraw-Hill Professional Publishing (2007)
3. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA (1995)
4. Vapnik, V.: Statistical Learning Theory, New York (1998)
5. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1-3) (2002) 389–422
6. Romano, R.A., Aragon, C.R., Ding, C.: Supernova recognition using support vector machines. In: ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications, Washington, DC, USA, IEEE Computer Society (2006) 77–82
7. Mukkamala, S., Janoski, G., Sung, A.: Intrusion detection: Support vector machines and neural networks. The IEEE Computer Society Student Magazine **10**(2) (2002)

8. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
9. Abdelnur, H.J., State, R., Festor, O.: KiF: a stateful SIP fuzzer. In: IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications, New York, NY, USA, ACM (2007) 47–56
10. Quittek, J., Niccolini, S., Tartarelli, S., Stiemerling, M., Brunner, M., Ewald, T.: Detecting SPIT calls by checking communication patterns. In: IEEE International Conference on Communications (ICC 2007). (Jun 2007)
11. Balasubramaniyan, V.A., Ahamad, M., Park, H.: CallRank: Combating SPIT using call duration, social networks and global reputation. In: Fourth Conference on Email and Anti-Spam (CEAS2007), Mountain View, California USA (2007)
12. Shin, D., Shim, C.: Progressive multi gray-leveling: A voice Spam protection algorithm. IEEE Network **20**
13. Yan, H., Sripanidkulchai, K., Zhang, H., Shae, Z.Y., Saha, D.: Incorporating active fingerprinting into SPIT prevention systems. In: Third annual security workshop (VSW'06), ACM Press (Jun 2006)
14. Reynolds, B., Ghosal, D.: Secure IP Telephony using Multi-layered Protection. In: Proceedings of The 10th Annual Network and Distributed System Security Symposium, San Diego, CA, USA (feb 2003)
15. Chen, E.: Detecting DoS attacks on SIP systems. In: Proceedings of 1st IEEE Workshop on VoIP Management and Security, San Diego, CA, USA (apr 2006) 53–58
16. Sengar, H., Wang, H., Wijesekera, D., Jajodia, S.: Detecting VoIP Floods using the Hellinger Distance. Transactions on Parallel and Distributed Systems : Accepted for future publication (sep 2007)
17. Valdes, A., Skinner, K.: Adaptive, model-based monitoring for cyber attack detection. In: RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, London, UK, Springer-Verlag (2000) 80–92
18. Denning, D.E.: An intrusion-detection model. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press (Apr 1986) 118–133
19. Krügel, C., Toth, T., Kirda, E.: Service specific anomaly detection for network intrusion detection. In: SAC '02: Proceedings of the 2002 ACM symposium on Applied computing, New York, NY, USA, ACM Press (2002) 201–208
20. Ning, P., Jajodia, S.: Intrusion Detection in Distributed Systems: An Abstraction-Based Approach. Springer (2003)
21. Maloof, M.: Machine Learning and Data Mining for Computer Security: Methods and Applications. Springer (2005)
22. Kang, H.J., Zhang, Z.L., Ranjan, S., Nucci, A.: Sip-based voip traffic behavior profiling and its applications. In: MineNet '07: Proceedings of the 3rd annual ACM workshop on Mining network data, New York, NY, USA, ACM (2007) 39–44
23. Nassar, M., State, R., Festor, O.: Intrusion detections mechanisms for VoIP applications. In: Third annual security workshop (VSW'06), ACM Press (Jun 2006)
24. Nassar, M., State, R., Festor, O.: VoIP honeypot architecture. In: Proc. of 10 th. IEEE/IFIP Symposium on Integrated Management. (Jun 2007)

Table 10. Appendix: List of features

Number	Name	Description
Group 1 - General Statistics		
1	Duration	Total time of the slice
2	NbReq	# of requests / Total # of messages
3	NbResp	# of responses / Total # of messages
4	NbSdp	# of messages carrying SDP / Total # of messages
5	AvInterReq	Average inter arrival of requests
6	AvInterResp	Average inter arrival of responses
7	AvInterSdp	Average inter arrival of messages carrying SDP bodies
Group 2 - Call-ID Based Statistics		
8	NbSess	# of different Call-IDs
9	AvDuration	Average duration of a Call-ID
10	NbSenders	# of different senders / Total # of Call-IDs
11	NbReceivers	# of different receivers / Total # of Call-IDs
12	AvMsg	Average # of messages per Call-ID
Group 3 - Dialogs Final State Distribution		
13	NbNOTACALL	# of NOTACALL/ Total # of Call-ID
14	NbCALLSET	# of CALLSET/ Total # of Call-ID
15	NbCANCELED	# of CANCELED/ Total # of Call-ID
16	NbREJECTED	# of REJECTED/ Total # of Call-ID
17	NbINCALL	# of INCALL/ Total # of Call-ID
18	NbCOMPLETED	# of COMPLETED/ Total # of Call-ID
19	NbRESIDUE	# of RESIDUE/ Total # of Call-ID
Group 4 - Requests Distribution		
20	NbInv	# of INVITE / Total # of requests
21	NbReg	# of REGISTER/ Total # of requests
22	NbBye	# of BYE/ Total # of requests
23	NbAck	# of ACK/ Total # of requests
24	NbCan	# of CANCEL/ Total # of requests
25	NbOpt	# of OPTIONS / Total # of requests
26	Nb Ref	# of REFER/ Total # of requests
27	NbSub	# of SUBSCRIBE/ Total # of requests
28	NbNot	# of NOTIFY/ Total # of requests
29	NbMes	# of MESSAGE/ Total # of requests
30	NbInf	# of INFO/ Total # of requests
31	NbPra	# of PRACK/ Total # of requests
32	NbUpd	# of UPDATE/ Total # of requests
Group 5 - Responses Distribution		
33	Nb1xx	# of Informational responses / Total # of responses
34	Nb2xx	# of Success responses / Total # of responses
35	Nb3xx	# of Redirection responses / Total # of responses
36	Nb4xx	# of Client error responses / Total # of responses
37	Nb5xx	# of Server error responses / Total # of responses
38	Nb6xx	# of Global error responses / Total # of responses