SS-IDS: Statistical Signature Based IDS

Payas Gupta, Chedy Raïssi, Gérard Dray, Pascal Poncelet, Johan Brissaud

SS-IDS: Statistical Signature based IDS

Payas Gupta
SIS, Singapore Management University
payas.gupta.2008@phdis.smu.edu.sg

Chedy Raïssi
LGI2P-Ecole des Mines d'Alès, France
Chedy.Raissi@ema.fr

Gerard Dray
LGI2P-Ecole des Mines d'Alès, France
Gerard.Dray@ema.fr

Pascal Poncelet
LGI2P-Ecole des Mines d'Alès, France
Pascal.Poncelet@ema.fr

Johan Brissaud
Beeware SA, Montepellier, France
jbrissaud@bee-ware.net

Abstract

Security of web servers has become a sensitive subject today. Prediction of normal and abnormal request is problematic due to large number of false alarms in many anomaly based Intrusion Detection Systems(IDS). SS-IDS derives automatically the parameter profiles from the analyzed data thereby generating the Statistical Signatures. Statistical Signatures are based on modeling of normal requests and their distribution value without explicit intervention. Several attributes are used to calculate the behavior of the legitimate request on the web server. SS-IDS is best suited for the newly installed web servers which doesn't have large number of requests in the data set to train the IDS and can be used on top of currently used signature based IDS like SNORT. Experiments conducted on real data sets have shown high accuracy up to 99.98% for predicting valid request as valid and false positive rate ranges from 3.82-7.84%.

1. Introduction

The deployment of computer systems and networks has greatly increased the risk of attacks on information system. For example, a recent study by the National Institute of Standards and Technology has shown that the damage to US companies, were estimated at more than 59.6 million dollars per Annum. Web servers are also very much prone to these attacks, as they are remotely accessible, often not properly configured and sometimes not updated.

Recently, many research groups have focused on generation of patterns automatically [2, 13]. Some IDS were built to detect intrusion to some extent in real time, but cannot be updated as many of them are constructed by manual encoding or some expert knowledge, changes to them are slow and expensive [1]. So there is an urgent and desperate need of a system which can learn automatically. Some of the recent researches show that they have started building up mechanisms for updating the IDS frequently and reducing the false alarms [10].

Our mechanism aims to build up the IDS based on *Statistical Signature* which can characterize and model the typical behavior on the web server [9]. The main motivation to build SS-IDS is to deal with the modified string of the old attacks which cannot be detected by the currently used IDS. As every web server has some *characteristic feature* about the kind of requests it receive, the central theme of our approach is to model those requests in some attributes and then automatically generate statistical signatures of the same which could be used to detect abnormal activities on the web server. Also, from few experiments this has been shown that SS-IDS is best suited for freshly installed web servers, which doesn't have a large number of requests in the data set for modeling the IDS.

The remainder of this article is organized as follows. Section 3 gives an overview of the framework. Section 4 discusses the algorithm behind the approach. The results obtained during different experiments are shown in Section 5. In Section 6 we conclude the paper and propose some future work.

2. Related Work

Many currently used IDS, like Snort works on misuse detection approach, but unfortunately it is hard to keep the sets of updated signatures with respect to continuously increasing vulnerabilities. So, anomaly detection method was used in building of many IDS. Our work has also a strong liaison with anomaly intrusion detection method used by [6]. The first anomaly based IDS was introduced in 1987, by D.E. Denning [3]. Some interest have already been expressed in this field. There are, however, differences in the approaches taken in the use of modeling the data.

In a paper by Wagner and DeanWe [12], which shows how the static analysis may be used to automatically derive a model of application behavior, this improved the approach [3] with high degree of automation, protection against a broad class of attacks based on corrupted code, and the elimination of false alarms. The first work on web based anomaly detection approach was done by extracting some parameters like length of the input data and then analyzing the behavior of the same [5]. But for doing the analysis, the system requires a high volume of data set for generating all possible grammar for testing, which clearly differentiate SS-IDS with their approach. A more recent work has investigated the potential of characterizing the normal requests in some attributes by Kruegel [6], used some models to distinguish valid requests from the invalid ones, but assumed the learning data set to be valid and contains no attacks. The limitations of anomaly-based intrusion detection system have been addressed by using both generalization and characterization techniques [11].

Researchers have also proposed solutions to deal with the problem of generating the real time IDS, but produce a high rate of false alarms [8]. Here we introduced a novel approach to mitigate the effect of false alarms produced in the anomaly detection method. SS-IDS is different in two ways i.e. *firstly*, it doesn't require a large no of request to model the system and *secondly*, it concentrate on the modified string of the old attacks i.e. the attack string is modified to bypass the currently deployed Web Server IDSs.

3. SS-IDS Description

Every web server has its own characteristics and inherently this may vary from one web server to another. Using this characteristic feature of the web server, our approach is based on the analysis of the HTTP requests (e.g. [4],RFC 1738 [7]) and model these requests in various attributes which could eventually generate signatures from it. The following example illustrates an entry in a simplified web server log. The */access.cgi* is the path and the other part of the request is the query which is composed of $< (user, jean) >$. We will use thereafter, the terms *Query* for the Query part of the request, Uri path(U) to describe the path (e.g. */access.cgi*) and Query argument value(Q) (e.g. *jean*) for the application portion.

```
192.233.57.105 - jean [15/Sep/2007:23:59:59 - 0800]
"GET /access.cgi?user=jean" 200 2123
```

3.1. Extraction of U and Q

From legitimate real data sets, each request is separated in *Uri* & *Query* part. We had assumed that all the requests in the training data set are valid requests and so, no matter whether it is a *Uri* or *Query*, all part of a request has to be valid. Therefore, calculating separately the values of each part shouldn't affect our final results and signatures as both parts are valid and contains no attack. The attribute values are calculated on the extracted *Uri path(U)*. However for the Query part of the request the values are calculated on each *Query argument(Q)*. Since the attack could occur on different parts of the request, therefore our goal is to separate each element to mitigate the effect of one by another. As in the *parameterization phase*, there is no liaison between U and Q of the request therefore they could be considered separately.

3.2. Classification of Attributes

The attributes i.e. the **typical feature** of a request are used to determine the activity on the web server. In fact, a selection of a particular attribute is important because it allows us to differentiate legitimate request and an attack request. Some attribute values are calculated on the original string and some on the decoded string. The string is decoded recursively until it is not fully decoded.

The attributes are specific to the characteristics of the web server and different tests were needed to retain the relevant ones. Below are the list and the reasons for our choice of some of the attributes chosen for the final experiments (the selection has been made with the assistance of an expert in the field of computer and network security).

1. **Number of %00 (NULL) characters** - is calculated on the fundamental string extracted from the U or Q of the request. This attribute is very efficient in detecting those attacks which use NULL character to ignore the pursuance of the rest of the string after the NULL character.

2. **Number of coded characters**

3. **Number of coded characters which wasn't requiring any encoding but was encoded**

4. **Number of coded characters which are encoded twice**

   Attributes 2-4 are calculated on the original string extracted from the U or Q. These attributes are very imperative as they characterize the encoding mechanism used by the user. In a normal scenario, the frequency of the number of encoded characters is very less in a request, unless there are some regional characters like ($\grave{e}$, $\hat{a}$, $\acute{e}$, $\tilde{o}$ etc.). Also, normally a user or web browser won't encode the characters which doesn't require any encoding like (*a-z*, *A-Z*, *0-9*) or they will not encode the character twice. This help us in detecting those attacks which encodes the attack part of the string and pass it as a parameter to the web server to bypass the IDS.

The other attributes are used primarily to count the numbers of **special characters**, **space**, **letters**, **numbers** ... in the string. These attributes are calculated on the decoded string. For example, for the following *URI* **/%63alendrier%27%3B%2A/../../** the number of coded characters in the U is 4.

3.3. Signature Generation and Analysis

After calculating the attributes' values, it was noticed that these values follow a normal probability distribution model with some standard deviation. So for each part of the request, the Mean (μ) and Standard Deviation (σ) for all the attributes were calculated separately for U and Q. Then for each of these, the value for the 1^{st} Standard Deviation is calculated.

After generating the distribution values for the real valid data sets, the same data set is taken and checked whether the attributes' values of each request lay in its own distribution or not. If the values of that attribute lies in its own interval, it is coded as **1** otherwise **0**. This form the vector bit signature for that valid request and this procedure is applied to all the requests in the data set. The unique signatures out of the all signatures are then used for the final testing. For efficiency reasons, we do not accept the final signatures obtained as the only unique encoded form of signatures as vectors bits, but we also have the provision for updating these signatures. With the help of these signatures, new requests on the web server could be tested to find misconduct, i.e. which do not match to all the signatures.

Signature generation is a very important step as it model the legitimate request into vector bits. It shows that some attributes' values of few requests didn't lie in its own interval of distribution, and they are coded as **0**, but it doesn't imply that the request is the invalid request. It is not necessary for the values of all attributes of a valid request to lay completely in its own interval i.e. the values which are outside the distribution are also important to us. These values cannot be ignored. The reason for generating signatures is very much important and distinguishes our approach with [6]. In [6], the approach generates an alarm only based on one parameter model and didn't take care of the correlation between the models, however we do take care of the correlation between the models by generating the *statistical signatures* which gives more flexibility and generates less false positives. The following example highlights it. Let us suppose that for a Query, the 1st attribute's value lie in the interim distribution and 2nd doesn't, so the signature generated by SS-IDS will be '**1 0**'. However the query is still valid and characterizes the behavior of the web server by showing that it can receive some queries whose 1st attribute's value lie in the distribution and value of the 2nd attribute don't. The approach by [6], will give an alarm generating false positive. SS-IDS also give importance and weight to those values which are outside the interval.

4. Proposed Algorithm

The general process is partitioned into two phases. In the *parameterization phase*, the IDS process valid requests and correspondingly generate signatures. The *testing phase* of the IDS examines the new request coming to the web server and separates the normal traffic from the anomalous. An alarm is triggered whenever it detects the new request as an attack.

4.1. Parametrization Phase

Performance of the IDS in the test phase is based on how the model of the IDS is constructed and the kind of the data sets used. The algorithm currently deals with only two fields i.e. U and Q but it could also be extended to all the fields of the header part of the request. Assuming the all petitions L of valid set of requests R. μ_i and σ_i are the averages (μ) and standard deviations (σ) for ith attribute of all the queries of the stage setting phase. To adapt to a principle of setting *on-line/real-time*, the values μ_i and σ_i are obtained incrementally, using Equations 1 & 2.

$$\mu_1 = x_1, \mu_{k+1} = \frac{k}{k+1}\mu_k + \frac{1}{k+1}x_{k+1} \qquad (1)$$

$$\sigma_1 = 0, \sigma_{k+1} = \sqrt{\frac{k}{k+1}\sigma_k^2 + \frac{k}{(k+1)^2}(\mu_k - x_{k+1})^2} \qquad (2)$$

Every time a new request is processed the corresponding attributes' values are calculated which further leads to change in μ and σ. After calculating the new μ, σ and the attributes' values for all the requests R in the valid data set L, the signatures for both U and Q are generated separately.

4.2. Testing Phase

In the learning phase, the U and the Query part of the same request were processed separately but in the testing phase, the whole request is processed as one. In this phase Function 1 will extract U and Q from the new request and then calculate the attributes' values. If the attributes' values lie in its own interval i.e. (μ-σ $\leq$ val $\leq$ μ+σ) which is calculated in the stage setting phase of the IDS, the attribute's value is coded as 1 otherwise 0. This process is repeated for all the attributes, and this forms the signature for both U and Q.

```
Function 1 Testing_Phase
  Data: Testing of Request $R$.
  Result: Alert for queries attacks OR disabled
  begin
      compute att. values for U and Q
      if $\exists$ $U$ then
          compute sig. $NR\_signature$ for U
      $w_{min} \longleftarrow 1$
      while $\forall$ sig. $S \in U$ signature file do
          $w \longleftarrow NR\_signature$ && $S$
          if $(w/no\_of\_attributes) < w_{min}$ then
              $w_{min} \longleftarrow w/no\_of\_attributes$
          if $w = 0$ then break
      $w_{max} \longleftarrow 0$; $w_{min} \longleftarrow 1$
      if $\exists$ $Q$ then
          compute signature $NR\_sig$ for Q
      while $\forall$ $Q$ do
          while $\forall$ sig. $S \in Q$ signature file do
              $w \longleftarrow NR\_signature$ && $S$
              if $(w/no\_of\_attributes) < w_{min}$ then
                  $w_{min} \longleftarrow w/no\_of\_attributes$
              if $w = 0$ then break
          if $w_{max} < w_{min}$ then $w_{max} \longleftarrow w_{min}$
      if $w_{min}$ of $U \neq 0$ OR $w_{max}$ of Query $\neq 0$ then
          alert Attack
      else
          $Valid$
  end
```

In both U and Q, the new signature is matched with other signatures of U and Q respectively which were obtained in the parametrization phase. For each signature the numbers of non-matching bits are counted, and the total sum is divided by the number of attributes of U. It is called the weight ($0 \leq$ weight ≤ 1) given to the new request by one signature. The process is repeated with all the signatures present in their corresponding files, and finally the minimum of all the weights is calculated. In case of U, this minimum weight shows how much the new signature of U is different from the maximum matching statistical signature. However in case of Query part of the request, this procedure is repeated for each Q and in each case the maximum of all the minimum weights calculated, shows how much the new query is different from the maximum matching statistical signature.

To illustrate why we use the maximum of all minimum weight in the Query component, let us consider the following case, suppose that there are three *Query arguments* in the new request, where the first two arguments are valid and the latter constitute an attack. Then for each of the arguments, we look for the most nearest match with other signatures and calculate their corresponding weight. Therefore, for the first two arguments it would weigh **weight=0** (for the exact match) and for the third argument, it would weigh some **weight⟩0**. Now the maximum of all the weights will give us the idea of the exact part where the attack had actually taken place. So weight of the query will become the maximum of all the weights of Q. If the request doesn't consitute any query part then there is no weight given for it and is left blank. For each request if there is some **weight⟩0**, then it reviewed that the complete request didn't match with the signatures which were generated in the stage-setting phase of IDS and it could *possibly* be an attack.

5. Experimental Results

To evaluate the performance of IDS in terms of detection accuracy, certain experiments were performed. Our purpose is to show the novelty of this methodology, more specifically its accuracy of detecting *modified string of the old attacks.* Experiments were conducted on a collected 6 months of real data log from the web server of the Company Beeware and École des Mines d'Alès.

5.1. Detecting Accuracy of the IDS

1700000 requests from the valid data set of Beeware company was used. A total of Q=**504497** &

U=**1700000** from the data set were obtained. After processing each request from the data set signatures were calculated. Experiments were performed with the several different testing data sets from Beeware company and fig. 1 was generated. From Fig. 1 it could be easily reviewed that range of False Acceptance Rate **(FAR)** ranges from *3.25% - 7.66%* and False Rejection Rate **(FRR)** ranges from *.0034% - .008%*.

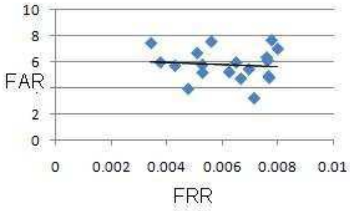

Figure 1. FAR vs FRR (Beeware Comp.)

The results were obtained without updating the signature set of IDS throughout the detection phase. The real data sets used during the testing phase of the system also contained some queries which could be easily detected by the currently used IDS (when searched manually). So integrating our approach with the traditional approaches can cover the entire set of attacks. As compared to other approaches our approach has a very high amount of accuracy of about 99.98% in true detection but not in terms of false negatives. This was due the fact that our approach is not good for detecting attacks like **/etc/passwd**, because the structure is same as **/abc/defghi**. Our approach aims on the behavioral analysis rather than the semantic analysis. As in the previous example, the request is syntactically same, so it leads to false alarm. But this could be easily handled by some currently used signature based IDS like SNORT. **SS-IDS** is resilient for those attacks which try to modify the attack string and evade into the system like **/%65%74%63/%70%61%73%73%77%64** which is just the hex encoding of the attack string.

The results were obtained without updating the signature set of IDS and the testing data set also contained some queries which could be easily detected by the currently available IDS.

5.2. Total requests required for training?

The motivation for this experiment was to calculate the number of requests to model a system. We have conducted this experiment because for a freshly installed web server there won't be many requests in the log for modeling the **system/web server**, but all anomaly based IDS requires a fair amount of web request for the feature extraction process. However by this experiment we have shown that, SS-IDS doesn't require the large data set for modeling the system.

In this experiment, randomly **20000** requests were chosen, i.e. about 2% of the entire data set for the training of the system. After learning one request, the IDS was tested with **3550** invalid requests and **7500** valid requests. Whenever a new request was learned by the system, the current signatures were updated and replaced by new set of signatures. Some graphs were obtained during this procedure as shown in Fig. 2. From the graphs it could be easily reviewed that the number of false alarms (i.e. the attacks which were detected as valid requests) after learning ≈ **7500** requests became constant. Nevertheless, the valid requests which were detected as **attack/illegitimate** traffic were ranging between from **18-38** requests after learning ≈ **12200** requests, which is acceptable in our case as our model is based on behavioral analysis. The experience with this experiment was very useful as it shows that in fact it is not mandatory to have the large amount of requests for apprenticeship. This approach is best for the freshly installed web servers, as it doesn't require many valid requests to model the system as compared to previous approaches which cannot be implemented without having the large data set of requests. The update procedure i.e. to update the entire database of the signatures requires just **6.52** seconds after learning **20000** requests. This can be improved by using high end processors. The experiment was conducted on *Intel Pentium 4* with *512 MB RAM.* SS-IDS could be very efficiently implemented in real time with some modifications in the system.

6. Conclusion

In this article, **SS-IDS** a new approach for detecting intrusion attempts on web servers was introduced using the log information generated by the web application. The proposed approach models each request based on some attributes and generates **Statistical Signatures** which are used to detect intrusion attempts. The novelty of this approach is its incremental way of updating the signatures, which doesn't require any specified value of threshold for attributes to classify attacks and non attacks. The approach has been developed to detect the modified string of the old attacks very efficiently. Some experiments have been conducted on the real data sets which clearly show the high accuracy of SS-IDS. Also, this has been shown from few experiments that SS-IDS is best suited for the freshly installed web servers, which doesn't have many requests

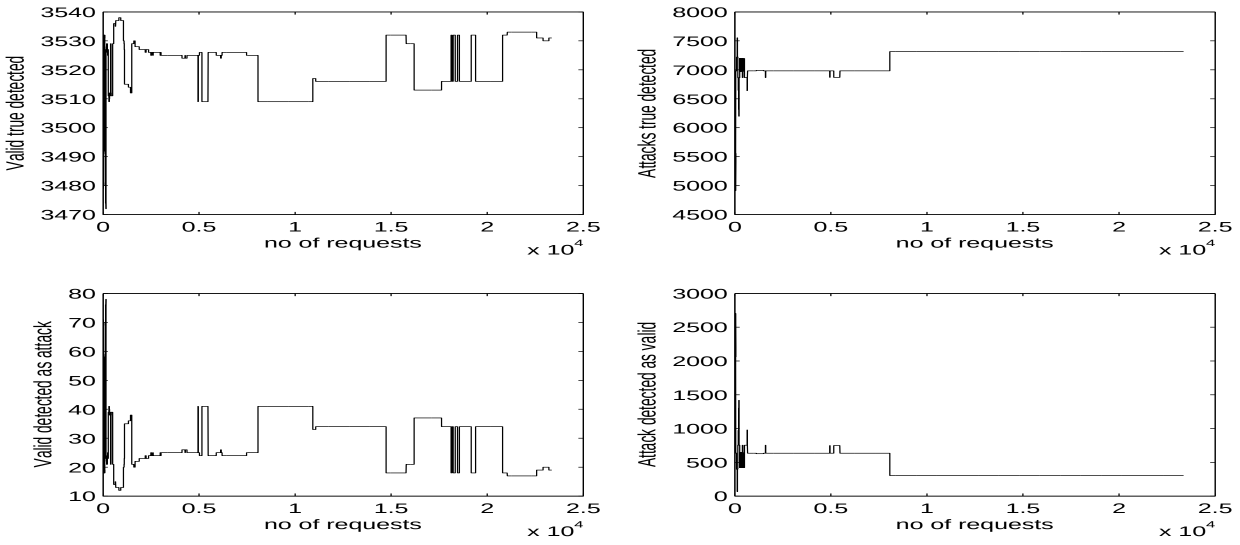

Figure 2. Updating the signatures after learning each requests

in the log for modeling the Intrusion detection system. SS-IDS can be used to make good classification by introducing structural information also.

7. Acknowledgements

This work was carried out under a project financed by the region of Languedoc Roussillon, resulting collaboration with the company BeeWare SA (http://www.bee-ware.net), which works in industrial results.

References

[1] J. J. G. Adeva and J. M. P. Atxa. Intrusion detection in web applications using text mining. *Engineering Applications of Artificial Intelligence*, 20(4):555–566, Jun 2007.

[2] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna. Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 63–86, Gold Coast, Australia, September 2007.

[3] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.

[4] T. A. S. Foundation. Apache http server documentation, 2007.

[5] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. *10th ACM Conference on Computer and Communications Security (CCS), ACM Press.*, Oct 2003.

[6] C. Kruegel, G. Vigna, and W. Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 48(5):717–738, Aug 2005.

[7] T. B. Lee, L. Masinter, and M. McCahill. Uniform resource locator, Dec 1994.

[8] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang. Real time data mining-based intrusion detection. *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings*, 1(1):89–100, Jun 2001.

[9] A. Pereira, G. Franco, L. Silva, and W. M. Jr. A hierarchical characterization of user behavior. *Proceedings of the WebMedia & LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress*, 00:2–9, 2007.

[10] T. Pietraszek and A. Tanner. Data mining and machine learning-towards reducing false positives in intrusion detection. *Elsevier Ltd*, 2005.

[11] W. Robertson, G. Vigna, C. Kruegel, and R. A. Kemmerer. Using generalization and characterization techniques in the anomaly-based detection of web attacks. *Proceedings of Network and Distributed System Security Symposium Conference, Internet Society, 2006*, Feb 2006.

[12] D. Wagner and D. Dean. Intrusion detection via static analysis. *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 156, 2001.

[13] D.-Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229–243, 2003.