Detecting anomalous Web server usage through mining access logs

Toni Gržinić
Croatian Academic and Research Network
National CERT
Josipa Marohnića 5, 10000 Zagreb, Croatia
toni.grzinic@foi.hr

Tonimir Kišasondi
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
tonimir.kisasondi@foi.hr

Josip Šaban
Hypo Alpe-Adria-Bank Internationl AG
Alpen-Adria-Platz 1, 9020 Klagenfurt, Austria
josip.saban@foi.hr

Abstract. *Most operating systems services generate log files that can be used for debugging and supervision. One important function of log files is logging security related or debug information, for example logging unsuccessful authentication or error logging. This paper shows how to implement an anomaly detection process of web server's unexpected events using the the Apache web server's logs and applying supervised machine learning algorithms to extracted features. Also, we compare the classification performance of several algorithms that can be easily implemented in real-world scenarios.*

Keywords. anomaly detection, access logs, Apache web server

1 Introduction

With the increase in the number of Internet users combined with malicious traffic flow forced organizations to prepare for the new and sophisticated threats. During the last few years the security industry rapidly grew and, as a consequence, the market was flooded with new firewalls, intrusion detection/prevention systems and other appliances which are helping to maintain the required level of organization's security. One of the main characteristics of modern web sites is the generation of high volume of data. While this increased data quantity is useful for organization's daily operations, it also holds valuable user information that can be used in the process of analysing security incidents.

Apache web server, the most popular platform used to serve web pages, stores information about users activity in human-readable log files. Collected data presents a valuable source of information that has to be properly archived and analysed. This problem is tackled by the upcoming part of the security market called Security information and event management (SIEM). The main characteristics of SIEM products are data aggregation and retention combined with correlation and alerting on specific events. But there are also concerns about commercial SIEM products and their closed-source algorithms - they mostly revolve about questions such as "how intelligent are they" or "can they learn and report new attacks". The problem with these types of applications is not in the data collection and archival part but in the domain of data correlation and statistical analysis; current solutions use basic statistical techniques relying on the intrusion analysts to detect possible problems.

Today, an emerging security threat is a distributed denial of service at application layer, where the attacker generates a vast amount of data causing the web server overload. This data is sent in HTTP requests in different varieties - malformed, legitimate or partial requests. Those packages are consumed by the web server, causing resources overload and making the web server unresponsive. It is also important to note that there are many robots scraping web sites, and attackers using web application scanners to find vulnerabilities in web sites.

To successfully detect these Internet threats, organizations must have an efficient and adaptable system that can distinguish normal and harmful traffic. In this paper we present and compare methods for detecting anomalous behaviour contained in web server log files by automating the detection of possible problems. This approach can help intrusion analysts in the process of detecting attacks or anomalous behaviour and reporting these findings to the competent authority.

2 Related work

Currently available tools usually use relatively simple methods for detecting anomalous behaviour. In order to demonstrate this approach a sample SQL injection query is presented from a compromised host:

```
192.168.25.35 - -
[28/Nov/2012:14:35:10 +0100]
"GET /c/main.php?cpage=buy&Page
    =511111111111111%22%20UNION%20SELECT%20
    CHAR(45,120,49,45,81,45),CHAR
    (45,120,50,45,81,45),CHAR
    (45,120,51,45,81,45),CHAR
    (45,120,52,45,81,45),CHAR
    (45,120,53,45,81,45),CHAR
    (45,120,54,45,81,45),CHAR
    (45,120,55,45,81,45),CHAR
    (45,120,56,45,81,45),CHAR
    (45,120,57,45,81,45),CHAR
    (45,120,49,48,45,81,45),CHAR
    (45,120,49,49,45,81,45),CHAR
    (45,120,49,50,45,81,45),CHAR
    (45,120,49,51,45,81,45),CHAR
    (45,120,49,52,45,81,45)%20--%20/*%20
    order%20by%20%22as%20/* HTTP/1.1" 200
    13494 "-" "Opera/9.00 (Windows NT 5.1;
     U; ru)"
```

To better illustrate three main approaches we use the following tools:

1. Log colorization tools which color simple classes of messages. For example, errors would be highlighted red, which helps to identify errors; two most popular colorizers are ATG Log Colorizer [6] and CCZE [5]. The weakness of this approach is that log colorizers do not remove the noise from all log content and they don't help the user identify what is actually happening. In this example the sample line would be a valid query, that is colorized as a valid query by the colorizer.

2. Probabilistic methods are those based on removing certainty from a set of log files. For example, volatile elements from a log file (IP addresses, timestamps, size of returned data, usernames, etc.) make it simple to identify certain attacks - we can take, for instance, a brute force attack that shows a single line with a large number of failed login attempts. Normal data (successful logins, requests for a single page, etc.) will also be represented with only one line, which results in the removal of large amount of data. Certain anomalous lines can be represented by a single entry in the entire reduced set of about 200 lines (sample taken from a few thousand line long log file) - in this example this sample line would be shown with a single entry and would be easily identified from a shorter detection file. This method is employed by tools like Petit [3].

3. Forwarding the current log file through a intrusion detection system (IDS) which can parse log files. This approach can be only applied to systems with IDS already present. The downside of this approach is that it is usually done post mortem, which defeats the security benefit of having an intrusion detection or prevention system.

Theoretical contributions in the field of information security address two main topics: differentiating benevolent and malicious users and detecting attacks based on the sent payload to the web server. There are also some research activities about web log mining that concern automatic web structure generation and link recommendations based on the users' usage patterns [16] [18].

Differentiating users based on the web site usage patterns is a main topic of detecting malicious web robots. This topic relies on meaningful feature extraction from web logs that are later used in data mining algorithms or statistical methods [10], [14] [15], [13]. Frequently used algorithms are Self-Organizing Map, Adaptive Resonance theory, C4.5 classifier, Bayesian networks, k-means clustering and many others.

Research topics that analyze payload sent to the web server can be divided in two categories - pattern orient and anomaly oriented. Meyer suggested an approach based on evaluating Apache access logs stored in common format [11]; recognition is based on POSIX or Perl compatible regular expressions that match malicious patterns of OWASP Top 10 attacks in requests made by clients. This approach is similar to signature based systems like Snort or Ossec.

Krueger and Vigna [9] also suggested an anomaly detection system which recognizes attacks from the requests' length, character distribution and n-grams which are later used in conjunction with a Hidden Markov model classifier. An extensive comparison of anomaly detection techniques can be found in Ingham PhD dissertation and work [8].

Foss et al [4] for instance, have shown the application of non-parametric clustering for the use in web log analysis. Their algorithm TURN, which is a non-parametric approach to mining log files, with various filters to improve clustering. TURN also allows efficient clustering without the need to specify parameters. Another algorithm they developed was ROCK, for the use in non-Euclidean spaces. This paper was written in 2001, and shows a continued trend in research of log file analysis. The methods presented in this paper were not implemented in any wide use tool that would help with log analysis.

3 Data sources

Apache web server generates huge amounts of useful information about processing errors and user's access. More precisely, the Apache web server logs all requests processed by the server, the location and content of the access logs is controlled by the directive CustomLog.

Log management process begins with the collection of access data using many available tools for crunching live and retrograded logs. Most popular freely available tools for this purpose are Webalizer, Analog, Piwik and Awstats, but it is also important to mention an interesting popular commercial software called Splunk [12] which can analyze different types of log files.

These tools parse log information and display number of visits, visits duration, authenticated users, page hits, domains and countries of host visitors, used operating system, used browser, worms activity, HTTP errors and many more. But this display of information does not solve the issue of anomalous and harmful content detection - there are solutions like IDSes, but those solutions rely only on signatures or static rules. One example of the state of the art IDS is Snort, which is a network IDS. Host-based Intrusion Detection Systems are used for analysing various system log files and user events, a popular example of host-based IDS is OSSEC.

A typical log entry in Apache access log file consists of the following:

```
66.249.66.134 - - [10/Jan/2012:11:13:01
    +0100] "GET /admin HTTP/1.1" 200 5715
    "-" "Mozilla/5.0 (compatible;
    Googlebot/2.1; +http://www.google.com/
    bot.html)"
```

The entry consist of:

- Client IP address (66.249.66.134)
- Timestamp and timezone of the request (10/Jan/2012:11:13:01 +0100)
- HTTP method used (GET)
- Path on the server (/admin)
- HTTP version (HTTP/1.1)
- Response code from the server (200)
- Size of the returned data, represented in bytes (5715)
- User agent field (Mozilla/5.0 (compatible; Googlebot/2.1;...)

Log entries are limited by the default configuration because they only store a part of the HTTP request and response.

In this paper we analysed the Apache web server access log files, using the fields containing IP address, requests, user agents, timestamps and HTTP status codes returned by the server. IP address can be used to acquire the owner information or whether it is used as a proxy, requests can contain attack vectors from specialized vulnerability framework dictionaries, user agent data reveals browser and operating system details. In some cases, it can also contain the name of the web scanner or the crawler/robot used. HTTP status codes reveal whether the request resulted as a successful response, redirection or caused an error.

4 Methodology

4.1 Used features and dataset labeling

The analyzed traffic dataset belongs to an internally used application, which was accidently open for inbound Internet connections and later used as a low interaction honeypot. The application was primarily used for record keeping, each month a system administrator logged in and inserted new records. Collected Apache web logs were suitable for analysis because they contained many web application attacks using unknown web scanners.

Dataset contains events that occurred throughout the year, from December 2010 until December 2011, with a total of 7300 events made by 849 different users.

Features used for detecting anomalous behaviour were: timestamp range, bandwidth usage, distinct IP addresses clustered by sessions, number of as autonomous system numbers (ASN) involved, total number of requests, percentage of error requests.

Due to the low traffic generation multiple actions were taken: the original dataset was split in parts with a time range of one day, care was taken in splitting various user sessions because broadband users' IP address change after 24 hours, every user session was identified by the same IP address and user agent in a span of one hour. Also, IP addresses were joined to ASNs which helped us to differentiate the users. To obtain error percentages we took requests with status codes between 400 and 500, which by the HTTP specifications are error codes.

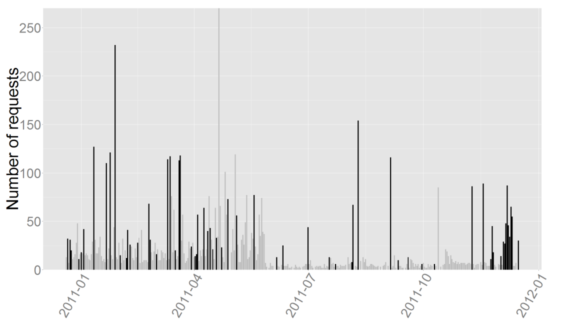

Figure 1: Web server's requested pages distribution from 16th December 2010 to 20th December 2011

The dataset was labeled using the interquartile range (IQR) outlier test method. That was made to avoid the assumption of data belonging to a Gaussian distribution, in our case IQR was used as a robust estimator. Then the dataset was skimmed through and wrong labels were corrected. IQR test accelerated the process of labeling the dataset; most errors caused by IQR test occurred because of wrong labeling values belonging to the lower boundary for outliers and some special cases.

In Fig. 1 we can see the request distribution trend in the observed time range; anomalous periods (labeled with the IQR method) are annotated in black. We can also observe that in our example high number of requests does not imply that the server was exposed to some type of error or attack, even though, in the previously mentioned application layer DDos and some other attacks, the attacker generates a fairly high number of requests.

4.2 Applied classification algorithms

After preprocessing raw web logs taken from the web server and extracting features needed for classification, several data mining algorithms were chosen for evaluation. We used algorithms from the Weka package [7]; all belong to the supervised learning category. The prerequisite for classification is that the basic dataset is split in two parts - one for training and the other one for testing the algorithm performance.

For the purpose of anomaly classification we used the following algorithms:

1. **C4.5 algorithm** [17] was developed by Ross Quinlan, as an extension to the present ID3 algorithm. C4.5 generates a decision tree classifier based on the concept of information entropy. Information gain is used for tree pruning, the pruning threshold used in Weka implementation *J48* is set to the confidence minimum of 25

2. **Random Forest** [1] is an ensemble method used in classification that fits multiple decision trees during the training phase and randomly chooses sub samples from these trees. This variant of bootstrapping aggregation is used to improve performance and prevent overfitting. The predicted class is usually an average or mode of randomly selected samples. In our experiments the forest consisted of 10 trees.

3. **Naive Bayes** is a simple probabilistic classifier which uses Bayes theorem with a "naive" assumption of independence. Although Naive Bayes can outperform more sophisticated classification algorithms, it can have a bad accuracy if the particular attribute is missing in the training set.

4. **Ripper** or Repeated Incremental Pruning to Produce Error Reduction is a propositional rule learner proposed by William W. Cohen. Ripper uses the divide and conquer approach to divide the dataset and constructs an overfitted (growing) tree iteratively pruning the tree until it achieves a satisfactory accuracy. The algorithm builds rules until it reaches a large error rate. The main difference between incremental reduced-error pruning and RIPPER is the optimization phase where RIPPER reconsiders rules and chooses the rule with the smallest description-length of rule set. Detailed description of the algorithm can be found in Cohen paper [2].

5. **Logistic regression** also known as multi response linear regression is usually used for binary classification. Logistic regression produces accurate probability estimates by maximizing the probability of the training data; in other words - it approximates a membership function. For every test instance the classifier calculates a membership to a class. The Weka algorithm *SimpleLogistic* uses LogitBoost for fitting the logistic model.

6. **Neural Network** is a feedforward artificial neural network (in Weka known as Multilayer perceptron) trained using the back-propagation method. The hidden layer consists of 4 neurons with sigmoid activation functions and learning rate α=0.3.

7. **K-nearest neighbours (k-NN)** is a non-parametric and instance-based method (in Weka know as *IB1*) for classifying objects based on closest examples in the feature space. If multiple features are the same smallest distance to the test instance, the first instance found is used. For our experiments we used k=1.

5 Results

We tested the classification performance of the selected algorithms using two different training datasets. Both sets were split from the original dataset, the first dataset used in Experiment 1 had 25% training instances and the second dataset used in Experiment 2 had 50% training instances. Results from Experiments 1 and 2 are shown in Table 1 and Table 2.

The results show that the Random forest algorithm has the best percentage of correctly classified anomalies in both Experiments.

Random forest, in Experiment 1, had the highest classification accuracy of 92%, the lowest classification accuracy belonged to the neural network (57%) and the k-nearest neighbours algorithm positioned itself in the middle (60%). Low precision rates from Experiment 1 were caused by the small size of the training set.

Best results, considering the area under the curve (AUC) metric, in Experiment 1, had Logistic regression (0.93) followed by the Random forest algorithm (0.92). It's interesting to see that some low performing algorithms in Experiment 1, like Logistic regression, k-Nearest Neighbours and Neural Network, despite the low percentage of correctly classified items, had a high recall value for detecting anomalies.

With the increased size of training set in Experiment 2 these algorithms performed better. In Experiment 2 Random forest model had the best performance with AUC 0.98, followed by the Neural Network (0.96), Logistic regression (0.94) and k-Nearest neighbours (0.9).

The anomaly detection process usually has a higher cost of false negatives (FN) than false positives (FP). FP give more work for the intrusion analysts but FN

Table 1: Experiment 1, results with 25% splitted dataset used for training

Algorithm	Correctly classified	Precision	Recall	AUC
C4.5	72.24%	0.39	0.81	0.84
Random Forest	92.54%	0.80	0.81	0.92
Naive Bayes	82.09%	0.52	0.66	0.84
Ripper	65.67%	0.30	0.61	0.69
Logistic Regression	77.61%	0.46	0.91	0.93
Neural Network	57.31%	0.30	0.89	0.86
k-NN	60.60%	0.31	0.89	0.71

Table 2: Experiment 2, results with 50% splitted dataset used for training

Algorithm	Correctly classified	Precision	Recall	AUC
C4.5	73.73	0.42	0.94	0.79
Random Forest	96.12	0.88	0.92	0.98
Naive Bayes	82.99	0.54	0.72	0.85
Ripper	91.64	0.76	0.81	0.87
Logistic Regression	93.13	0.82	0.83	0.94
Neural Network	94.63	0.93	0.78	0.96
k-NN	86.27	0.59	0.95	0.90

complicate the process of finding anomalies. Following this assumption we decided that the better model will have the highest recall rate and a reasonable (not too low) precision rate.

From the results it is evident that Random forest and Logistic regression are the best models for implementation. In case we plan to use a large training sample, we can consider the Neural network and the k-Nearest neighbours model.

6 Conclusion

Main problems in anomaly detection are that data does not follow a specific distribution (and, consequently, we can not use classic statistical approaches and methods) and that statistical properties of features used in the training process unexpectedly change over time.

The second mentioned issue is known as concept drift which is a phenomenon that can be avoided by updating the training model periodically. For example, after some time period, the model will be retrained with the recently observed values. It is also important in data streams to correctly determine the time window which will be used for learning. Our example consisted of a low bandwidth usage, thus the time window of one day was sufficient for most cases. Regardless ensemble method gives the best classification result, other simpler methods like Logistic regression and Neural network can perform with a comparable classification rate. Considering the training set size we can implement the simpler model trading off a small performance rate and reducing complexity in the model implementation.

As future work, we plan to improve the process of self-training using semi-supervised learning or robust statistic methods, which are not affected by outliers or need the assumption that the data belong to the normal distribution.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] William W. Cohen. Fast effective rule induction. In *In Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.

[3] CrunchTools.com. Petit log analysis. `http://crunchtools.com/software/petit/`, 2013. Accessed 30.4.2013.

[4] Andrew Foss, Weinan Wang, and Osmar R Zaïane. A non-parametric approach to web log analysis. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001)*, pages 41–50. Citeseer, 2001.

[5] freecode.com. Ccze log colorizer. `http://freecode.com/projects/ccze`, 2013. Accessed 30.4.2013.

[6] Kelly Goetsch. Atg log colorizer. `http://atglogcolorizer.sourceforge.net/`, 2013. Accessed 30.4.2013.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[8] Kenneth L. Ingham and Hajime Inoue. Comparing anomaly detection techniques for http. In *Proceedings of the 10th international conference on Recent advances in intrusion detection*, RAID'07, pages 42–62, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 251–261, New York, NY, USA, 2003. ACM.

[10] Haibin Liu and Vlado Kešelj. Combined mining of web server logs and web contents for classifying user navigation patterns and predicting user' future requests. *Data & Knowledge Engineering*, 61(2):304–330, 2007.

[11] Roger Meyer. Detecting attacks on web applications from log files. Technical report, SANS Institute, 2008.

[12] Adam Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Commun. ACM*, 55(2):55–61, February 2012.

[13] Athena Stassopoulou and Marios D Dikaiakos. Web robot detection: A probabilistic reasoning approach. *Computer Networks*, 53(3):265–278, 2009.

[14] Dusan Stevanovic, Natalija Vlajic, and Aijun An. Unsupervised clustering of web sessions to detect malicious and non-malicious website users. *Procedia Computer Science*, 5(0):123 – 131, 2011. The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011).

[15] Pang-Ning Tan and Vipin Kumar. Discovery of web robot sessions based on their navigational patterns. In *Intelligent Technologies for Information Analysis*, pages 193–222. Springer, 2004.

[16] Mislav Šimunić, Željko Hutinski, and Mirko Čubrilo. Log file analysis and creation of more intelligent web. *Journal of information and organizational sciences*, 29:25–39, 2005.

[17] Ian H Witten, Eibe Frank, and Mark A Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.

[18] YY Yao, HJ Hamilton, and Xuewei Wang. Pageprompter: an intelligent web agent created using data mining techniques. In *Rough Sets and Current Trends in Computing*, pages 506–513. Springer, 2002.