Detecting Web Attacks Using Multi-Stage Log Analysis

Melody Moh*, Santhosh Pininti, Sindhusha Doddapaneni, and Teng-Sheng Moh
Department of Computer Science
San Jose State University
San Jose, CA, USA
*Corresponding author; Email: melody.moh@sjsu.edu

***Abstract*— Web-based applications have gained universal acceptance in every sector of lives, including social, commercial, government, and academic communities. Even with the recent emergence of cloud technology, most of cloud applications are accessed and controlled through web interfaces. Web security has therefore continued to be fundamentally important and extremely challenging. One major security issue of web applications is SQL-injection attacks. Most existing solutions for detecting these attacks use log analysis, and employ either pattern matching or machine learning methods. Pattern matching methods can be effective, dynamic; they however cannot detect new kinds of attacks. Supervised machine learning methods can detect new attacks, yet they need to rely on an off-line training phase. This work proposes a multi-stage log analysis architecture, which combines both pattern matching and supervised machine learning methods. It uses logs generated by the application during attacks to effectively detect attacks and to help preventing future attacks. The architecture is described in detail; a proof-of-concept prototype is implemented and hosted on Amazon AWS, using Kibana for pattern matching and Bayes Net for machine learning. It is evaluated on 10,000 logs for detecting SQL injection attacks. Experiment results show that the two-stage system has combined the advantages of both systems, and has substantially improved the detection accuracy. The proposed work is significant in advancing web securities, while the multi-stage log analysis concept would be highly applicable to many intrusion detection applications.**

Keywords—Log Analysis, Pattern Matching, SQL injection, Supervised Machine Learning, Text Classification, Web Security.

I. INTRODUCTION

Web-based applications have become prevalent in the ubiquitously-connected world. The recent increase in demand for huge data storage and high processing speed has led to the era of cloud computing, yet, most cloud-based systems and applications are accessed through the Internet web. Web security therefore has remained as top priority for Internet- and Cloud-Services Providers (ISP/CSP).

In order to improve ISP/CSP user confidence and to protect web-based systems and applications, strong, effective security mechanisms must be deployed. Many architecture designs have been proposed and many cryptographic algorithms implemented; yet intruders continue to gain access to web-based application, to steal confidential information, and to make unwanted modifications – including recent intrusions made to Target, the Home Depot, BlueCross Insurance, and many hospital and government information systems. SQL injection is one major attack that has been happening to web-based applications [3].

Log analysis is one major procedure to detect when an intruder attacked the system, how it happened, and what steps were performed during the attack [6]. The traditional approach of manual log analysis is no longer possible for real-time log analysis, as the number of logs has rapidly increased. Manual analysis also involves more cost and much more time. Technologies such as Hadoop and Hive therefore have been used.

Two major techniques have been used in log analysis: pattern matching and machine learning [10] [2]. While the pattern matching method may work dynamically, only known patterns can be recognized, yet new types of injections may be created when only small changes are made to existing patterns. Machine learning also has its limitation, since classification in machine learning algorithms works with probabilities, it may not be able to correctly classify SQL injections that combine groups of words each was classified with high probability as non-SQL injection.

Most existing log analysis methods for SQL injection detection are based on either pattern matching or machine learning. We propose an effective multi-stage log analysis architecture that uses both methods. The system therefore combines both of their advantages and compensates for their disadvantages.

The major contributions of this paper may be summarized as follows:

- Proposed a multi-stage architecture for detecting web attacks, using SQL injection attacks as an example.
- Implemented a prototype based on the proposed architecture, using Bayes Net and Kibana.
- Careful compared the pros and cons of pattern matching (Kibana) and machine learning (Bayes Net) methods.
- Evaluated the 2-stage system through a series of experiments. Detailed discussion of SQL injections detected and undetected by each of the four evaluated methods.

The rest of the paper is organized as follows: Sections 2 and 3 review background and related work, respectively. Section 4 presents the proposed system design and prototype implementation. Section 5 illustrates experiments and results. Finally, Section 6 concludes the paper.

978-1-4673-8286-1/16 $31.00 © 2016 IEEE
DOI 10.1109/IACC.2016.141

II. Background and Related Studies

In this section, background knowledge in SQL injection, log analysis, pattern matching and machine learning methods are briefly described. Due to page limitation, related studies have been omitted.

A. SQL Injection:

This is one of the major attacks made to web applications. Attacker inputs an SQL query, which modifies or damages the database that is connected to the target web application. SQL injections are broadly classified into three types based on the methods involved in creating them [14]. They are: *(1) Order* Wise, which involves execution of code written by the hacker so that it provides unlimited and unauthorized access to the database; *2) Blind*, in this case an attacker gets access by asking a series of questions to the database and obtains a complete idea of the database structure based on the answers; *3) Against Database,* in which an attacker exploits the input validation vulnerabilities to create an SQL query so as to discover the desired information.

B. Log Analysis

Log Analysis is a process of understanding logs and extracting useful information. One of the open-source logging frameworks is Log4j [6], which is developed in Java by Apache Software Foundation. It has been used in this work to log information such as timestamp, log level, error messages, and user information such as IP address, username, and requested URL. Once these logs are captured, log analysis is then performed to do further investigation on website users.

C. Pattern Matching

One of the techniques used in this work to detect SQL injections is pattern matching. It checks whether a set of words is present in the given text. One commonly used pattern-matching methods are described below.

1) ELK (Elasticsearch, Logstash, and Kibana)

This is a complete system that includes collection, search, analysis and visualization of data. The system essentially solves some of the main problems encountered when using traditional databases, including inconsistent data and time formats, and lack of visualization [24]. Its major components are described below.

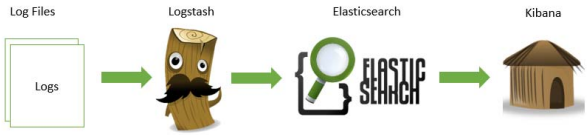

Fig 1: Logstash, Elasticsearch and Kibana [28]

Logstash: It is a data pipelining tool that connects to a variety of sources with the help of plugins, and streams data to an analytics system [24]. Logstash receives different types of logs, namely system logs, web server logs, error logs, and application logs. These are normally distributed among different systems using different formats. Logstash helps users to parse data into one single common format before storing into the analytics data store. Additionally, Logstash provides a way to parse custom format logs by providing custom logics.

Elasticsearch: It is an open-source search and data analysis software which gives users a deep insight on streaming data [24]. This tool provides a scalability feature by allowing users to add new nodes. Once a cluster is set up, Elasticsearch provides search and analysis features by building a distributed environment on top of Apache Lucene, which is used for full-text searching.

Apache Lucene: It is an open-source text search engine library written in Java [30]. It allows users to write their own queries through its query API, which helps users to search GeoIP locations, perform multilingual searches, etc. It also provides different types of searches such as term and phrase searches, and allows users to group keywords for detailed text searching.

Kibana: It is an open-source data visualization interface for real-time summarizing and charting of stream data [26]. It helps users understand large volumes of datasets by providing different visualizations like bar charts, pie charts, line spots, and maps. It also provides different visualizations that can be combined into custom dashboards.

D. Machine Learning:

Machine learning is a way of making a computer learn and take action without explicitly programming it [7]. It has been used in many areas such as big data search, spam filtering, etc. It may be broadly classified into supervised learning and unsupervised learning.

1) Naïve Bayes Classification

It is a simple probabilistic classifier. It builds upon the Bayes theorem, which gives the probability of an event occurring based on the given conditions that are related to the event. One major limitation is its assumption of independence between the attributes [27]; i.e., the existence of one attribute does not affect the other.

2) Bayes Net Classification

One problem in using a Naïve Bayes classifier is its assumption of treating all attributes as strongly independent (i.e. Probabilistic independence) of each other. This assumption seems unrealistic as it cannot be applied in situations where correlation exists between these attributes and unwarranted data needs to be ignored to improve performance.

Bayesian Networks are directed acyclic graphs (DAG) that represent the joint probability distribution over a set of random variables in a problem domain. Each variable lies at every vertex in the graph and the edges from the correlations between these random variables [1]. The conditional independence between these variables is stated in a way that each variable is independent of its non-descendants, given the state of their parent variables. Bayesian Networks are often used to tackle the independent-attributes assumption of Native Bayes Classification, and is helpful and improves performance. It has been used, along with other supervised machine learning methods, on our recent work on sentimental analysis [31].

III. System Design and Implementation

The proposed system is a two-stage intrusion detection system. It performs log analysis to protect a web-based application from SQL-injection attacks. It uses both machine learning and pattern matching techniques. For the clarity of description, Bayes Net and Kibana (of ELK) are used to represent machine learning and pattern matching methods, respectively. They are also used in the proof-of-concept prototype implementation.

The proposed architecture consists of three main parts: (1) *Log Generation*: this includes a logging system such as Log4j, (2) *Data Preprocessing*: it handles preprocessing needed for machine learning and pattern matching methods, and (3) Detection Methods: such as Bayes net for machine learning and Kibana for pattern matching.

In the following, for clarity, a simple, single-stage architecture is first illustrated. It is followed by the proposed multi-stage architecture and its description.

A. Single-Stage Architecture:

In this architecture, as shown in Figure 2, the application logs are generated using a logging library such as Log4j. Then, either machine learning method (follow the path on the right) or pattern matching method (the left path) is used for detecting SQL injection attacks. Preprocessing is needed for the machine learning method, which uses WEKA and then Bayes Net as an example. The pattern matching method is shown using the ELK system. The produced results are then presented to the analyst for further understanding and interpreting. The details of each step are described in the next subsection, on the proposed multi-stage architecture.

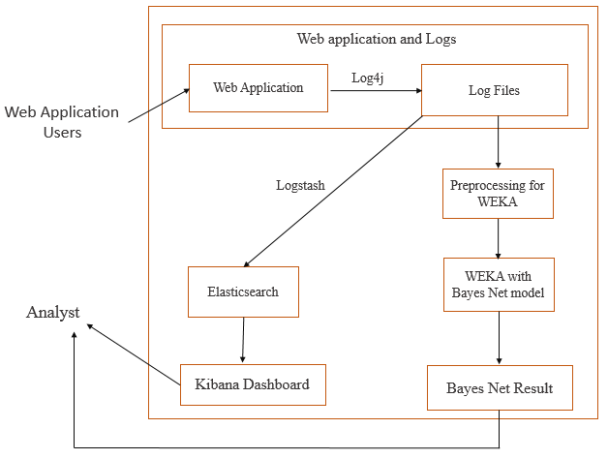

Fig 2: Simple architecture (1-stage)

B. Proposed Multi-Stage Architecture

The proposed architecture combines both machine learning and pattern matching methods to improve the detection strength. As shown in Figure 3, the Bayes Net results from machine-learning method is further fed to the pattern matching method. Alternatively, the pattern-matching method may be applied before machine learning. Detailed steps are described below.

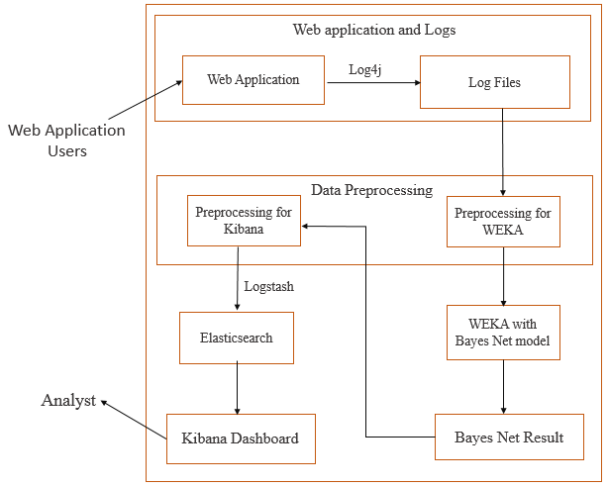

Fig 3. Proposed Architecture (2 stage)

1) Offline Training for Machine Learning:

Recall that off-line training is needed for supervised machine learning. WEKA is used in the proposed system, in which log files, with an added class attribute and labeled logs, are loaded. The training data is preprocessed where unnecessary attributes are removed. Furthermore, the following steps may be applied to further improve the training process.

Stemmers: A stemmer trims the suffix of the words so that the given word will be converted to its root word. This reduces the dictionary size and increases the efficiency of the model.

Stop words: These are words that do not play any role in the classification. Their removal yields better results. Since some of the simple words like 'and' and 'or' do play a major role in SQL injections, the default stop word list cannot be used. Instead, a customized stop word list is needed to yield good results.

Cross validation: It is a technique used while creating the model to reduce the estimation variance. In K-fold cross validation, the whole dataset is divided into K parts, where K-1 parts are used for training and the remaining part is used for testing. This procedure is done repeatedly K times until all the parts are tested. In this project, 5-fold cross validation is used. We achieved a detection rate of 78.8% for the BayesNet model with a standard deviation of 0.9.

2) Log Generation:

Consider a large number of users distributed around the world, it would be difficult to manually check the system-generated logs that consist of many unnecessary logs. What is needed are application logs. In order to generate application logs, one may use parsers and filters to filter out the unnecessary information in the logs, or use logging libraries (such as Log4j) to create custom logs.

3) Preprocessing:

There are two different kinds of pre-processing, one for WEKA and the other for Kibana; described below.

Preprocessing for WEKA: As attributes in the test set should match those in the training set, a class attribute and the required header are added, and the test data converted into the ARFF format before loading into WEKA. All the unnecessary attributes are also removed to match the model.

Preprocessing for WEKA (for 2-stage architecture): In this case, logs not detected by Kibana are the input for WEKA. As the current version of Kibana has not supported exportation of results from a query directly, it is first necessary to create a visualization that represents these logs not detected by Kibana. These results are then downloaded in the CSV format. A Unix script may then be used to convert the CSV file into an ARFF file for WEKA input. This is followed by the usual WEKA preprocessing, as described above.

Preprocessing for Kibana (for 2-stage): For the single-stage architecture, no pre-processing is needed. For the two-stage architecture, the output of WEKA is used as the input of Kibana. Thus, WEKA result needs to be "preprocessed" for Kibana. A Unix script may be used to make the necessary changes such as replacing commas with tabs, removing the header part of the ARFF file, and converting it into a text file. Then, Logstash will take the data from the text file and load it in Elasticsearch, after which Kibana will display the results.

4) SQL Injection Detection:

For the one-stage architecture described above, two different results are obtained, using Bayes Net and using Kibana. For the two-stage architecture, results of one method (Bayes Net model for machine learning) are fed into the other method (Kibana for pattern matching). Note that the order of the two methods may be swapped. Using Kibana, a custom dashboard may be easily created, showing many interesting characteristics about the detected injections.

IV. Experiments and Results

This section discusses dataset, experiment setup, results and discussions.

A. Experiment Setup

1) Dataset

The datasets used for these experiments were web application logs generated using the Log4j framework. The total number of logs used for the experiment was 12,000, as summarized in Table 1.

Table 1: Data Set

Data	*Total Logs*	*SQL Logs*	*Regular Logs*
Training Set	2000	547	1453
Testing Set	10000	2812	7188

2) The Web Application

As a proof-of-concept prototype, a web application used for a company internal discussions has been developed, using Java, Bootstrap, HTML, CSS, and JavaScript. MySQL was used as the database server. This web application is hosted in the Amazon AWS Linux instance, which offers a flexible, scalable, experiment environment [22].

3) *Kibana and Bayes Net*

As mentioned above, Bayes Net and Kibana (of ELK) are used to represent machine learning and pattern matching methods, respectively. These two methods are compared in Table 2 below, which may serve as a typical comparison between machine learning and pattern matching methods.

Table 2: Comparison Between Kibana and Bayes Net

Kibana	*Bayes Net*
Purpose	
Used for detecting SQL injections and visualizing data.	Used for classification of logs into SQL-related logs and other logs.
Mechanism	
Use pattern matching techniques for detection.	Use supervised machine learning to learn and detect attacks.
Overhead	
No file conversion is required; it takes data directly from text files.	Load only ARFF file, so log files need to convert to ARFF format.
No preprocessing is required; can use filters to extract only the required data.	Preprocessing is required; need to preprocess data manually before giving to model for classification.
No training is required; queries are written for detection.	Training is required, which involves manual classification that is time-consuming.
Pros and Cons	
A real-time system where new queries may be issued, and changes may be seen visually as soon as a logs are generated.	Not a real-time system as it involves an offline training phase.
Can detect only specified patterns, so cannot detect new types of SQL injections.	Can detect new patterns since it considers other attributes like IP address while classifying.
Results are in visualized form; easier to analyze.	Results in text format; more difficult to analyze.

B. Single-Stage Results:

The results obtained by the one-stage simple architecture are presented below.

1) Pattern Matching:

Using Kibana, the pattern matching method results in **85.3%** accuracy in detecting the SQL injections. Some visualizations are shown below.

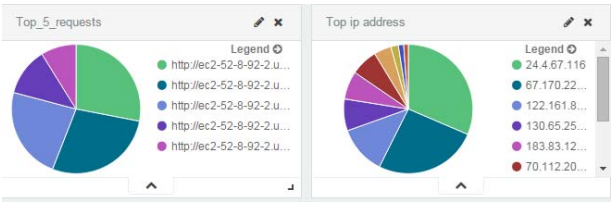

Fig 4: Top requested URLs, top user IPs

Note that the accuracy result depends on the queries. The following is a simple example that illustrates the challenge of writing a good query: In order to find SQL injections with SELECT statements, one can write a Kibana query to find the logs that have both "select" and "from". Yet, in general, there are many cases where a log consists of both words but it is not an SQL injection.

2) Machine Learning:

First, the Naïve Bayes method was used, from which 61.7% accuracy was obtained. This was clearly due to the fact that the attributes of these logs are not independent.

When Bayes Net method was used, the following confusion matrix, shown in Table 3, was generated by WEKA for the given test set. Note that, since only SQL injection detections are needed, we consider only positives for calculating accuracy. Thus, the accuracy result equals to **80.04%** or 2251/(2251+561).

Table 3: Confusion Matrix

True Positive	*False Positive*	*True Negative*	*False Negative*
2251	561	7132	56

There are a few obvious reasons that machine learning method such as Bayes Net does not obtain high accuracy. One obvious case is similar to the one stated above; i.e., words which that are in regular logs might also exist in SQL injection logs and vice versa. The example of "Select" and "From" serves this purpose. They are two regular words, but in combination might also create an SQL injection. In Bayes Net classification, one cannot specify the exact format so the results might be wrong. Also, since Bayes Net uses all the attributes, some of the new SQL injections might not be detected; take, for example, a new type of SQL injection from a previously normal user.

C. Two-Stages Results:

In this, we discuss the results obtained by combining pattern matching and machine learning methods.

1) Pattern Matching followed by Machine Learning

In this case, Kibana first catches SQL injections by pattern matching, then Bayes Net model catches SQL injections that were not detected by Kibana. This combination has achieved **94.7 %** accuracy.

SQL injections that were not detected by Kibana but subsequently detected by Bayes Net may include the following: (1) Some user names may contain SQL injection patterns such as SQL keywords, special characters, and wildcard characters. In Kibana, Lucene queries are used for pattern recognition, but Lucene cannot identify special characters such as #, <, <=, >, < in the username field. Since Bayes Net also uses other attributes for classification, it can successfully detect these injection logs. (2) Since the grouping concept was used in queries to retrieve meaningful information from the logs, some SQL injection patterns that contain only individual keywords rather than combinations might not be detected by Kibana but were detected by Bayes Net. Consider the following example:

*Select * from table* (detected in Kibana and Bayes Net)

Select table (detected in Bayes Net but not in Kibana)

2) Machine Learning followed by Pattern Matching:

In this case, the Bayes Net model first classifies logs, and then Kibana detects more SQL injections based on queries and classification (done by Bayes Net model). The result is a high accuracy of 95.4%. The Kibana visualizations in Figures 5, 6, and 7 show some details related to detected SQL injections, including a global map showing all the detected attacks, a zoomed-in view of the map, and the IP addresses, the most used URL and the log level of these detected attacks.

Fig 5: Map showing SQL injections

Fig 6: Zoomed view of Map showing SQL injections

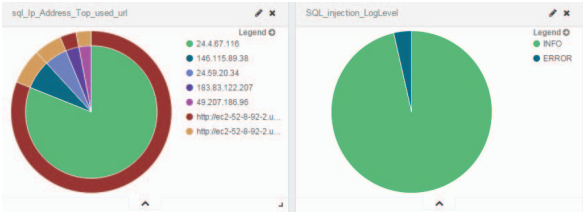

Fig 7: IP addresses with SQL injections and their most used URL, Log levels of SQL injections.

Reasons that SQL injections not detected by Bayes Net are detected by Kibana may include the following: (1) Log level, username and other attributes might be contained in regular logs in the training set. Since Bayes Net classifier considers also relations apart from individual attributes, it might not have detected SQL injection in some cases. (2) Since Bayes Net classifies logs based on probabilities of all the words present in a given log, it may not detect logs that contain a large number of words belonging to a regular log category. (3) It cannot detect new types of SQL injections that have words not contained in the training set. If the other attributes in the given log belong to the regular log category, Bayes Net may wrongly classify this type of logs as a regular log.

D. Overall Result Comparison

Table 4 shows the accuracies obtained by all the above experiments. It is clear that the proposed two-stage architecture is able to obtain results that are far more accurate than those from one-stage architecture. Note that the current best accuracy of 95.4% may further be improved by increasing the training set size and by using more sophisticated queries.

TABLE 4. RESULTS

Method	***Accuracy for SQL Detection (%)***
Machine Learning: Naïve Bayes	61.7
Machine Learning: Bayes Net	80.0
Patten Matching: Kibana	85.3
Kibana followed by Bayes Net	94.7
Bayes Net followed by Kibana	95.4

V. CONCLUSION

With numerous data flowing through the web every day, it remains important to detect SQL injection attacks that cause severe security problems on any web-based application hosted on the Internet or on the cloud. A multi-stage log analysis architecture has been proposed, which uses both machine learning and pattern matching methods. Experiment results have shown that the two-stage system is able to detect significantly more SQL injections that a single-stage system. When Bayes Net model (a supervised machine learning method) precedes Kibana (a pattern matching system), the combined system has achieved the best result. Since Kibana provides the final output with visualization, it is easy for analysts to further understand, interpret, and take further actions. Future work may include further improvement on Kibana queries, using other machine learning methods to reduce manual log classification efforts, and using

unsupervised learning methods which may lead to a real-time multi-stage log analysis system.

REFERENCES

[1] R.R Bouckaert, “Bayesian network classifiers in weka,” Department of Computer Science, University of Waikato, 2004.

[2] G. Buja, K. Bin Abd Jalil, F. Bt Hj Mohd Ali, & T.F.A. Rahman, "Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack," In Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on (pp. 60-64), April 2014.

[3] G.T. Buehrer, B.W. Weide, & P.A.G. Sivilotti, “Using parse tree validation to prevent SQL injection attacks,” In Proceedings of the 5th international workshop on Software engineering and middleware (pp. 106-113), September 2005.

[4] E. Dogbe, R. Millham, & P. Singh, “A combined approach to prevent SQL Injection Attacks,” In Science and Information Conference (SAI), 2013 (pp. 406-410), October 2013.

[5] K. Garcia, R. Monroy, L. Trejo, C. Mex-Perera, & E. Aguirre, “Analyzing Log Files for Postmortem Intrusion Detection,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 1690-1704, 2012.

[6] C. Gülcü, & S. Stark, “The complete log4j manual,” 2003.

[7] A. Joshi, & V. Geetha, “SQL Injection detection using machine learning,” 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014.

[8] T. Kalamatianos, K. Kontogiannis, P. Matthews, “Domain Independent Event Analysis for Log Data Reduction,” Computer Software and Applications Conference (COMPSAC), 2012.

[9] M. Kumar, M.Hanumanthappa, “Scalable Intrusion Detection Systems Log Analysis using Cloud Computing Infrastructure,” Computational Intelligence and Computing Research (ICCIC), IEEE International Conference, 2013.

[10] P. Kumar, & R.K. Pateriya, “A Survey on SQL injection attacks, detection and prevention techniques,” In Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on (pp. 1-5), July 2012.

[11] A. Makanju, A. Zincir-Heywood, & E. Milios, “A Lightweight Algorithm for Message Type Extraction in System Application Logs,” IEEE Transactions on Knowledge and Data Engineering, 2012.

[12] D. Marinescu, “Cloud computing theory and practice,” Boston: Morgan Kaufmann, 2013.

[13] A. Sadeghian, M. Zamani, & A. Manaf, “A Taxonomy of SQL Injection Detection and Prevention Techniques,” 2013 International Conference on Informatics and Creative Multimedia, 2013.

[14] C. Sharma, & S. Jain, “Analysis and classification of SQL injection vulnerabilities and attacks on web applications,” 2014 International Conference on Advances in Engineering & Technology Research, (ICAETR - 2014).

[15] X. Shu, J. Smiy, D. Yao, & H. Lin, "Massive distributed and parallel log analysis for organizational security," In Globecom Workshops (GC Wkshps), 2013 IEEE (pp. 194-199), December 2013.

[16] K.W. Ullah, A.S. Ahmed, J. Ylitalo, “Towards Building an Automated Security Compliance Tool for the Cloud,” Trust, Security and Privacy in Computing and Communications (TrustCom), 12th IEEE International Conference, 2013.

[17] X. Wang, Z. Zhang, M. Wang, L. Zu, Z. Lu, & J. Wu, “CDCAS: A Novel Cloud Data Center Security Auditing System,” Services Computing (SCC), IEEE International Conference, 2014.

[18] J.J. Wiley, F.P. Coyle, “Semantic Hedgehog for Log Analysis,” Internet Technology and Secured Transactions, International Conference IEEE, 2012.

[19] T. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, & E. Kirda, (2013) “Beehive: Large scale log analysis for Detecting suspicious activity in enterprise networks,” ACSAC, 2013.

[20] E. Yoon, & A. Squicciarini, "Toward detecting compromised mapreduce workers through log analysis," In Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on (pp. 41-50), 2014.

[21] M. Zolotukhin, T. Hamalainen, T. Kokkonen, & J. Siltanen, “Analysis of HTTP Requests for Anomaly Detection of Web Attacks,” 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, 2014.

[22] AWS “Amazon Elastic Compute Cloud (EC2) -Scalable Cloud Hosting,” Available: http://aws.amazon.com/ec2/

[23] “CloudComputingSecurity,”
Available: https://en.wikipedia.org/wiki/Cloud_computing_security

[24] "Elasticsearch: RESTful, Distributed Search & Analytics | Elastic," Available: https://www.elastic.co/products/elasticsearch

[25] “How to Set up the ELK Stack- Elasticsearch, Logstash and Kibana,” Available:http://knowm.org/how-to-set-up-the-elk-stack-elasticsearch-logstash-and-kibana/

[26] "Kibana: Explore, Visualize, Discover Data | Elastic," Available: https://www.elastic.co/products/kibana

[27] “Naïve Bayes”
Available: http://scikit-learn.org/stable/modules/naive_bayes.html

[28] “NetEye: Integration Logstash/Elasticsearch/Kibana,” Available: http://www.neteye-blog.com/2014/10/neteye-integration-logstashelasticsearchkibana/

[29] “Weka 3: Data Mining Software in Java,”
Available: https://www.cs.waikato.ac.nz/ml/weka

[30] "Welcome to Apache Lucene,"
Available: https://lucene.apache.org/

[31] M. Moh, A. Gajjala, S. Gangireddy, and T.-S. Moh, “On Multi-Tier Sentiment Analysis using Supervised Machine Learning,” Proc. IEEE/WIC/ACM Web Intelligence Conference,” Singapore, Dec. 2015