***Southeast Asian J. of Sciences: Vol 7, No 2,*** (2019) pp. 115-132

# ANOMALY DETECTION SYSTEM OF WEB ACCESS USING USER BEHAVIOR FEATURES

**Pham Hoang Duy, Nguyen Thi Thanh Thuy**
and
**Nguyen Ngoc Diep**

*Department of Information Technology*
*Posts and Telecommunications Institute of Technology (PTIT)*
*Hanoi, Vietnam*
*duyph@ptit.edu.vn; thuyntt@ptit.edu.vn; diepnguyenngoc@ptit.edu.vn*

**Abstract**

The growth, accessibility of the Internet and the explosion of personal computing devices have made applications on the web growing robustly, especially for e-commerce and public services. Unfortunately, the vulnerabilities of these web services also increased rapidly. This leads to the need of monitoring the users accesses to these services to distinguish abnormal and malicious behaviors from the log data in order to ensure the quality of these web services as well as their safety. This work presents methods to build and develop a rule-based systems allowing services' administrators to detect abnormal and malicious accesses to their web services from web logs. The proposed method investigates characteristics of user behaviors in the form of HTTP requests and extracts efficient features to precisely detect abnormal accesses. Furthermore, this report proposes a way to collect and build datasets for applying machine learning techniques to generate detection rules automatically. The anomaly detection system of was tested and evaluated its performance on 4 different web sites with approximately one million log lines per day.

**Key words:** Anomaly detection system, web log, rule generation, user behavior, TF-IDF.
2010 AMS Mathematics classification: 91D10, 68U115, 68U35, 68M14, 68M115, 68T99.

# 1 Introduction

Anomaly detection can refer to the problem of finding patterns in the data that do not match the expected behavior. These nonconforming patterns are often referred to as anomalies, exceptions, contradictory observations, and irregularities depending on the characteristics of different application domains. With the development of the Internet and web applications, anomaly detection in web services can range from detecting misuse to malicious intent which degrade the quality of website service or commit fraudulent behaviors. With web services, analytic techniques need to transform the original raw data into an appropriate form that describes the session information or the amount of time a user interacts with the services provided by the website.

In monitoring users' access to web services, a rule-based anomaly detection technique is commonly used due to its accessibility and readability to services administrators. There are two basic approaches to generating rules. The former is based on a rule manually and statically created by service administrators when analyzing users' behaviors. Another approach is to dynamically and automatically create rules using data mining techniques or machine learning.

For static rule generation, it is first necessary to construct a scenario of the situation that the administrator wants to simulate. For example, if there is one process running on one device and another process running on another device at the same time and the combination of both processes causes a security issue, the administrator needs to model this scenario. Besides creating these rules, administrators must enforce the correlation among these rules to verify whether the case is an anomaly or not. A rule can contain many parameters such as time frame, repeating pattern, service type, port, etc. The algorithm then checks the data from the log files and finds out attack scenarios or unusual behaviors.

The advantage of this approach is the ability to detect anomaly behaviors of access by correlating analysis and thus detecting intruders difficult to detect. There are specific languages that allow the creation of rules as well as tools to create rules effectively and easily. For companies with appropriate resources and budget, it is easier and more convenient to buy a set of rules than to use several systems to create specific rules.

The downside of this approach is high cost, especially for maintaining the set of rules. Modeling each attack scenario is not an easy and trivial task. It is most likely to re-perform the same type of attack and sometimes the anomaly cannot be identified. In addition, attack patterns can change and new attack forms are invented every day. As such, it is necessary to evolve the set of rules over time despite the fact that there is the possibility that some unspecified attacks, that could easily occur, are unrecognized.

The dynamic rule-generation approach has been used for anomaly detection for quite some time. The generated rules are usually in the form of if-then.

First, the algorithm generates patterns that can be further processed into a set of rules allowing to determine which action should be taken. Methods based on dynamic rule generation can solve the problem of continually updating attack patterns (or rules) by looking for potential unknown attacks. The disadvantage of this approach is the complexity of multi-dimensional spatial data analysis. Algorithms capable of processing such data often have high computational complexity. Therefore, it is necessary to reduce the dimension of the data as much as possible.

Our work proposes the development of anomaly detection system for web services based on dynamically generating rules using machine learning techniques. The data used in the analysis process is log entries from web services. In particular, Section 2 presents research on the use of machine learning techniques for the generation of anomaly detection rules. Section 3 describes the proposed anomaly detection system of web access as well as how to collect and build the dataset for building model that automatically generates detection rules. At the end of the work, the evaluation of the proposed system in term of execution time and detection performance and future work are presented.

## 2 RELATED WORK

Rule-based anomaly detection techniques learn rules representing the normal behaviors of users. The case of checking not being followed by any rule is considered an anomaly. These types of anomaly detection techniques may be based on classifiers using machine learning techniques that operate on the following general hypothesis: *The classifier can distinguish between normal and abnormal classes that can be learned in certain characteristic spaces.*

The multi-class classification assumes that training data contains labeled items belonging to common multi-class grades as in [1] and [2]. Such anomalous detection techniques rely on a classifier to distinguish between each normal label and the other. A data item being examined is considered abnormal if it is not classified as normal by any classifier or set of rules corresponding to that class. Some classification techniques combine reliable scores with their predictions. If no classifier is reliable enough to classify the data item as normal then this data item is considered abnormal.

The rule-based technique for a multi-class problem basically consists of two steps. The first step is to learn the rules from the training data using algorithms such as decision trees, random forests, etc. Each rule has a corresponding confidence that this value is proportional to the importance of the case. The second step is to find the best representation rule for the test samples. Several variants of rule based techniques have been described in [3, 4, 5, 6, 7].

Association rule mining [8] has been applied to detect anomalies in a single-class pattern by creating rules from unsupervised data. Association rules are

created from a categorized dataset. To ensure that the rules are closely linked to the patterns, support thresholds can be used to eliminate rules with low support levels. Association rule mining techniques have been used to detect network intrusion behavior as in [9, 10, 11].

FARM (Fuzzy Association Rule Model) was developed by Chan et al. [12] to target SOAP or XML attacks against web services. Most research on anomaly detection systems on servers and networks can only detect low-level attacks of the network while web applications operate at a higher application level. FARM is an anomaly detection system for network security issues especially for web-based e-commerce applications.

A number of anomaly detection methods for web service intrusion detection system are called Sensor web IDS to deal with abnormal behavior on the web [13]. This system applies algorithms based on theories of medians and standard deviations that are used to calculate the maximum length of input parameters with URIs. To detect misuse and abnormal behaviors, the proposed model uses Apriori algorithm to exploit a list of commonly used parameters for URIs and determine the order of these commonly used parameter sequences. The found rules will be removed according to the parameter length used in the maximum allowed URIs based on the calculation of median value in the dataset. This model incorporates association rule mining techniques and a variety of data sources including log data and network data that allow detection of various types of misuse as well as unusual behaviors related to attacks like denial of service, SQL injection.

Detecting anomalies using rule generation techniques with multi-class classification can use powerful algorithms to distinguish cases of different classes. This allows identifying in detail groups of normal as well as unusual behaviors. On the other hand, the verification phase of this technique is often very fast because each test sample is compared with the previous calculation model. With the formulation of rules from labeled dataset, the accuracy of labels assigned to different classes (typically normal and abnormal) has a decisive effect on the performance of the rule set, which is often difficult in practice.

# 3 PROPOSED METHOD

## 3.1 Classification model for detecting anomalies

Classification problem, one of the basic problems of machine learning, in order to learn the classification model from a set of labeled data (training phase), then, classify the test sample into one of the classes by using the learned model (verification phase). As introduced in the previous section, with the detection of abnormal behaviors of machine learning techniques, it facilitates the construction of classifiers, automatically learning the characteristics of each class to classify, such as intrusive and normal behaviors by learning from sam-

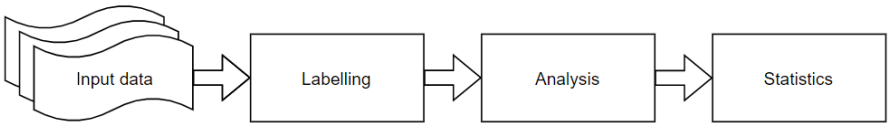


Figure 1: Steps for data processing

ple data. This approach allows for greater automation in case of new threats such as modifying old attack techniques but retaining some characteristics of previous hacking.

In order to apply machine learning techniques to classify abnormal or normal, one must first build labeled datasets for training. The basic steps in data processing are shown in Figure 1. Each record in the dataset describes features and a label (also called a class). These characteristics are derived from certain characteristics of user behaviors, such as the size of the query or the frequency of a certain parameter segment in the query; label is a binary value indicating whether the query is normal or not. Analysis to identify characteristics of user behavior can apply basic techniques such as identifying structures or components in the collected data. Statistical analyzes add user behavioral characteristics such as representations of interoperability between data components or abstract representations of collected data structures.

With the detection of web service anomalies, statistical and analytical techniques can be applied to a user's query string and transforming the strings into simple statistical characteristics such as the number of elements in the retrieval. For example, the parameters, how to use these components, or how the component correlates with abnormal and normal user behavior. Figure 2 shows two main stages in applying the classification model for anomaly detection.

The Random Forest algorithm [14] is used in the training process of the classification machine learning model. This popular algorithm allows a rule set to classify the input data. Random Forest [14] is a classification and regression method based on combining the predicted results of a large number of decision trees. Using a decision tree is like an election where only one person votes. Generating decision trees from a data sample to diversify the "votes" for conclusions. The application of techniques to generate data samples or a random selection of branches will create "malformed" trees in the forest. The more types, the more votes will provide us with a multi-dimensional, more detailed view and thus the conclusions will be more accurate and closer to reality. In fact, Random Forest has become a reliable tool for data analysis.

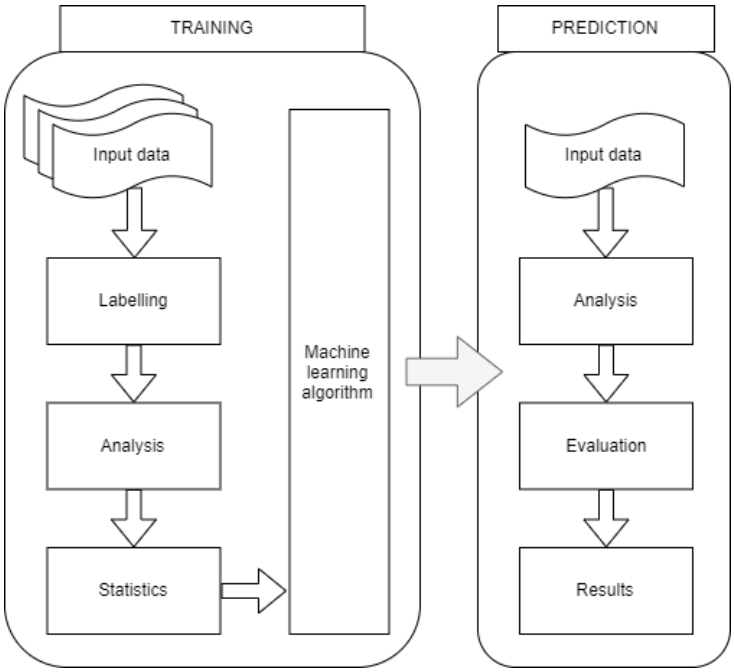


Figure 2: Phases of classification model

## 3.2 Representation of user behavior data

User interactions with web services are stored in server log files. The key information in these files is the web service queries encapsulated by HTTP protocol. The header of HTTP queries is used to extract information for training the detection model. Furthermore, these queries are required to be labeled as normal or abnormal. For example, first the URL (http://host/users) is extracted and paired with the HTTP method (e.g. GET, POST, PUT, etc.). On the other hand, HTTP queries also contain parameters for web services, for example *parameter1=value1&parameter2=value2*. These parameters should also be extracted and modified appropriately according to the machine learning model used. The following section examines some ways of representing user behaviors including common and TF-IDF features from the collected data.

### 3.2.1 Common features

Each HTTP query can be represented with common features [15] based on simple statistics about the parameters being sent, structural statistics in parameters as well as paths (URIs). The follow represents these features. Firstly,

statistical characteristics of a query include:

- Query length
- Length of the parameters
- Length of station information description
- Length of the "Accept-Encoding" header
- Length of the "Accept-Language" header
- Length of header "Content-Length"
- Length of the "User-Agent" header
- Minimum byte value in the query
- Maximum byte value in the query

And, characteristic of parameters sent to the server:

- Number of parameters
- Number of words in the parameter
- Number of other characters in the parameter

Characteristics of links to web pages:

- Number of digits in the path to the page
- Number of other characters in the path to the page
- Number of letters in the path to the page
- Number of special characters in the path to the page
- Number of keywords in the link to the page
- Path length

Statistical measures adds information about parameters which support detecting associations between features as well as quantifying the correlation with the type of user behaviors that the machine learning model wants to distinguish. The use of all or part of the above features has firstly impact on the performance of the classification algorithm. In other words, the analysis quality depends directly on the feature selection used in the model that represents the behavior of user in the query.

### 3.2.2 TF-IDF features

On the other hand, the sequence of parameters in HTTP query can also be encoded using the TF-IDF measure on keywords or key phrases in parameters. The TF-IDF measure is a common measure in text analysis [16], which indicates the frequency of the occurrence of the keyword TF (Term Frequency) and its inverse IDF (Inverse Document Frequency). TF and IDF measurements are determined as follows:

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) : w \in d\}} \tag{1}$$

$$idf(t,D) = log\frac{|D|}{|\{d \in D : t \in d\}|} \tag{2}$$

where *f(t,d)*: number of occurrences of keyword $t$ in the user's query parameter; $max\{f(w,d) : w \in d\}$ : the most occurrences of the keyword $w$ in the query; $|D|$: total user query parameters; $|\{d \in D : t \in d\}|$: number of documents containing $t$.

This TF-IDF measure facilitates the evaluation the similarity between HTTP queries. With data from the web log, parameters used in queries are separated from the content of the original query, followed by markers (such as '=') in the query parameter. To determine the TF-IDF measure, it is common for the sequence of parameters to be converted into 3-word phrases (n-gram = 3) and carry out the TF-IDF determination for these 3-word phrases.

## 3.3 Evaluation

The representation of user behavior to web service has a direct and significant effect on the performance of the machine learning model, consequently, detection of abnormal behavior. The HTTP CSIC 2010 dataset [17] was used to evaluate the effectiveness of user behavior representation with Random Forest learning.

HTTP CSIC 2010 dataset contains traffic generated targeting e-commerce web applications. In this web application, users can purchase items using the shopping cart and register by providing some personal information. The dataset is automatically created and contains 36,000 common requests and more than 25,000 abnormal requests. HTTP requests are labeled as normal or abnormal, and the dataset includes attacks like SQL injection, buffer overflows, crawling, file disclosure, CRLF, XSS injection, and parameter modification.

To evaluate the results of the classifier for detecting unusual behavior in user queries, the dataset was divided into two training and verification parts. The ratio is divided into 70% for training and 30% for verification. The results of the abnormal query classification of specific representations of user web service queries are shown in Table 1.

Table 1: Accuracy of anomaly detection methods

| Methods | Accuracy (%) |
|---|---|
| Common features with RF | 95.59 |
| TF-IDF features with RF | 99.50 |

The results in Table 1 show that the selection of user query representation has a great influence on the accuracy of anomaly detection. TF-IDF representation method is significantly more accurate than using common features. However, the time and complexity to process data according to TF-IDF is likely larger than those of common features. Therefore, when the amount of data increases significantly, this could be a problem to TD-IDF method. Within the scope of this work, TF-IDF was selected for machine learning model because of its accuracy and compared against the common features.

# 4 Rule based anomaly detection system of web access

Formulating a rule set for anomaly detection can be done by applying a machine learning model to detect anomalies through a decision tree algorithm. Decision trees are structured hierarchies that allow classification of data into different classes according to different branches by the rules. In other words, these rules show the relationship between data characteristics and classified classes. In case of necessity, decision trees can decay into discrete rules.

In the anomaly detection problem, data on user behaviors need to be divided into normal and abnormal. The set of decision tree rules can determine the class of user behavior based on the characteristic of user behavior.

The following section present in details classification models applied to anomaly detection based on random forest algorithms.

## 4.1 System description

Figure 3 shows the steps in the training phase to build anomaly detection model suitable for the collected data. The first step, preprocessing, is to transform the information of the user's service query to a format that conforms to the chosen algorithm. Basically, this step removes unnecessary information and selects the appropriate features.

The next step is to select the features. The remaining information of the user query is transformed into a data format suitable for training. The simplest form is the common data statistics such as the average value of the query string, the number of parameters in the log record, etc. This step can use

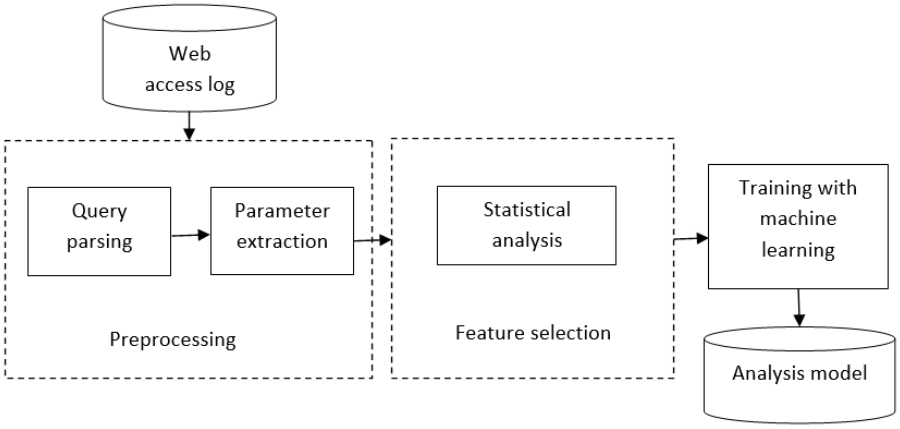


Figure 3: Steps of building an anomaly detection model

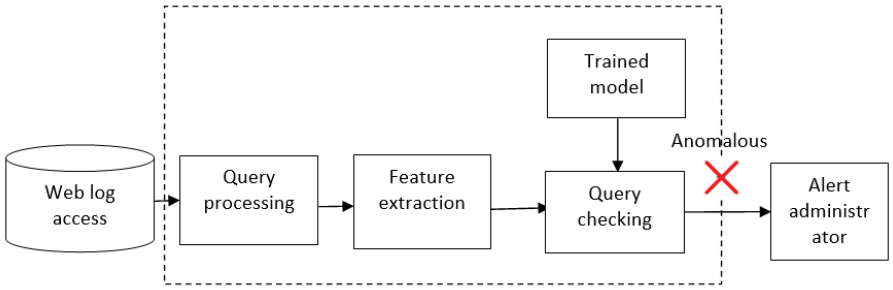


Figure 4: Steps of anomaly detection

more statistical analysis to transform character string data to numeric value for training process.

The final step is training to use Random Forest algorithms to create training models. The selected model is saved to serve for later detection of abnormal access. Detecting abnormal user behavior to alert the administrator is done through the steps shown in Figure 4.

Similar to the analysis model building phase, when a user accesses to the web service, user queries are read from the log files and modified in accordance with the anomaly detection model. Next is calculating the characteristics that represent these queries to obtain the best performance. Examination of anomalous queries of users is done by using analytical model obtained from the training phase. In abnormal cases, the administrators are alerted via the messaging module.

## 4.2 System development

The Python programming environment and scikit-learn library suite provide a diverse set of tools that allow rapid and efficient deployment of machine learning models. On the other hand, this development environment also allows convenient connection with database management systems suitable for managing large volume of web log data such as MongoDB. The proposed system was developed based on Python 3.6, the scikit-learn library, and Mongodb database management system. It performs following functions:

- Stores data about the user's web service query. The information about the query to the user's web service is transferred from the log file (Web log) to the standard format and stored in the MongoDB database for retrieval and processing.
- Convert web log data into TF-IDF and common features that can be used for building classification models. Representing the user's access to the web service has a decisive influence on the performance of the anomaly detection model. As surveyed and evaluated in the introduction, TF-IDF is the best way to represent user access.
- Develop classification models using machine learning techniques with random forest algorithms. The random forest algorithm is an advanced decision tree algorithm that improves the learning performance of the classification model. As such, this algorithm allows the generation of rules to effectively detect abnormal access. The learned model will be saved for later detection of anomalies.
- Detecting abnormal behavior. This phase is similar to the model building stage except that it is using the learned model to find out which users' access to the web service abnormal. The analytical results are saved for further investigation of the administrators.
- Warning. Notify the administrator or responsible person for unusual behavior through a predetermined channel. The current system is limited to email communication.

## 4.3 Building datasets

HTTP CSIC 2010 dataset is a common sample dataset used in the evaluation and testing of the performance of anomaly access detection models in the field of research. Although, the dataset contains anomaly access patterns in various forms, such as XSS attacks, SQL injections, applying this for analyzing user access behavior of a specific web service may not really fit. Therefore, building appropriate datasets for web services plays a decisive role in maintaining performance of systems as well as analyzing abnormal behavior of users. This work

proposes a semi-automated way to build a dataset for building classification models.

Security assessment tools for web services provide important sources of abnormal access patterns in various forms such as hijacking, XSS, or SQL injection. These templates are very useful for building sample datasets. However, these templates are not widely provided and the format is not entirely consistent with the end-user web service access analysis system. This work uses security assessment tools to generate abnormal access patterns which can be used for building model. In addition to the abnormal access patterns, normal ones are generated automatically by the web service structure scanning tool. For dynamic web sites, the scanning process is manually supported by the administrator. Thus, the system will be provided with 2 sample data access files including abnormal pattern and normal pattern.

OWASP Zed Attack Proxy (ZAP) [18] is one of the most popular open source security tools and is actively maintained by a large user community. ZAP can help web service administrators to automatically find security issues, especially during the development and testing of applications.

In this work, ZAP is used to generate abnormal access queries to web services that the administrators want to monitor. ZAP is set to maximum operating mode to obtain the most types of access (hijacking, XSS, SQL injection, etc.) as well as the maximum number of generated samples. These samples are saved as semi-structured files for later processing (e.g.: anomaly.csv). At the same time, ZAP generates reports that show specific security issues of the interested web services. These will be collected and saved in a separate file (e.g.: blacklist.csv).

In addition to security analysis tools, ZAP provides a mechanism for scanning web service structures through web spider and AJAX spider services. These tools allow to collect information about the structure of service pages and save them in the text file (e.g.: normal.csv). This work conducted using ZAP tool to collect data of web service access to 4 test websites with the following volume: about 300,000 abnormal queries, 200,000 normal accesses, about 6,000 web pages with security issues (blacklist).

In addition to the data generated through the ZAP toolkit, log data to four test web sites is also used to generate the dataset. Access by users with an HTTP code outside of the normal range (return code $> 200$) will be considered an abnormal access due to an error. In addition, accesses in the log files coincide with those of anomaly.csv and blacklist.csv files, which are considered abnormal. The way to identify unusual access from the log file is not entirely straightforward, but it is still useful because from an administrative point of view, access to the faulty web service should be alerted.

Data from the web log files from 4 test sites was sampled over 4 to 5 days with each site. The data collected included nearly 340,000 normal and nearly 56,000 abnormal accesses. Among the test sites, 1 site has an average data

volume significantly lower (about 25%) compared to the remaining sites.

The data collected from the log file combined with the data generated from the ZAP toolkit, after eliminating duplicates, constructs the dataset for building classification model. In fact, this dataset is quite balanced, including nearly 470,000 normal and 380,000 abnormal accesses.

## 4.4 Experiment and evaluation

### 4.4.1 Performance experiment

As mentioned above, an anomaly detection system for web access was developed based on Python 3.6 and scikit-learn library. The dataset from the above is divided in the ration of 7 : 3 for training and testing corresponding to the number of accesses $554,000 : 238,000$. These accesses are represented by TF-IDF to perform machine learning techniques using random tree learning. The parameters used in the classification model building process are set to the default level. Test results show that the accuracy of classification model is about 95%. Table 2 details the classification of normal and anomaly accesses from a learned model.

Table 2: Confusion matrix

| | **Normal** | **Abnormal** |
|---|---|---|
| **Normal** | 139,167 | 3,039 |
| **Abnormal** | 8,311 | 105,578 |

The results obtained from building classification models are relatively good. The classification model obtained from the model building step was used on the access data obtained from the log file to produce results as shown in Table 3.

The data in the table shows that at some point the number of user access doubles the average at other times. The number of abnormal fluctuations is not proportional to the access volume of the user. Figure 5 shows the percentage of hits checked (which do not coincide with other accesses) and anomalies in the number of queries tested. During the observed period, the number of anomalies fluctuated around 2% of the total number of queries examined. Particularly, in the $3^{rd}$ and $4^{th}$ observation range, the rate of abnormalities increased sharply.

Several abnormal user accesses are shown in Table 4. Accesses from 1 to 4 are relatively clear acts of attack on web services. Access 5 and 6 are not really obvious behaviors or attempts to browse the directory structure of a website. However, these behaviors still need to be monitored by administrators.

Table 3: Anomaly detection results

| **Total access** | **Total check** | **Abnormal** |
|---|---|---|
| 3,759,770 | 39,311 | 238 |
| 917,004 | 7,989 | 102,445 |
| 1,486,577 | 30,884 | 2,718 |
| 682,988 | 25,936 | 1,291 |
| 1,640,803 | 32,530 | 262 |
| 1,356,999 | 30,716 | 147 |
| 1,007,064 | 27,002 | 60 |
| 1,864,161 | 30,051 | 74 |
| 2,434,863 | 35,028 | 204 |
| 1,788,273 | 30,232 | 458 |
| 1,895,390 | 44,230 | 116 |
| 1,553,298 | 38,047 | 90 |

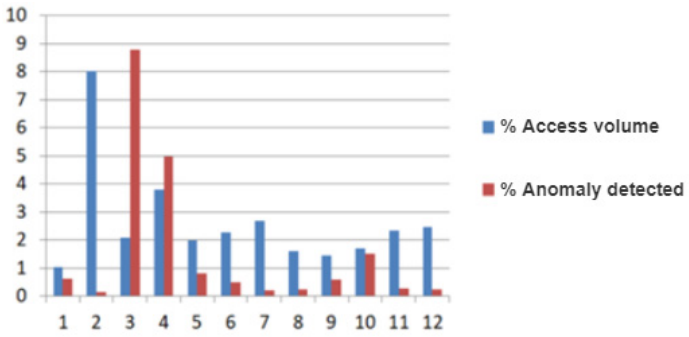


Figure 5: Percentage of prediction results

### 4.4.2 Run-time experiment

Despite the better detection quality of TF-IDF model when applying against CSIC data-set, this model perform is equally good compared to common-feature model with accuracy of 95.57% and 96.01% respectively with the data-set supported by ZAP tool. For training phase using the same machine and data-set, it takes almost 7 minutes to construct TF-IDF model compared about a half of minute to build common-feature model. Therefore, it is more important and interesting to investigate the run-time of these two models during testing phase (detecting anomaly).

These models, namely TF-IDF and common-feature models, are used to detect the anomaly against the datasets collected in 8 different days and each

Table 4: Abnormal access

| No. | Abnormal access |
|---|---|
| 1 | /Default.aspx?sname=..%2f..%2f..%2fetc%2fpasswd&sid=1293&pageid=32306 |
| 2 | /Default.aspx?sname=http%3a%2f%2fwww.google.com+&sid=1293&pageid=32306 |
| 3 | /wps/wcm/connect/309b0a0042eaedc58881ccd8919db02e/HINHLON.bmp?MOD=AJPERES |
| 4 | /Default.aspx?sname=c%3A%2FWindows%2Fsystem.ini&sid=4&pageid=468 |
| 5 | /public/upload_nhieuanh/server/php/_index.php |
| 6 | /vanban.aspx?type=%2527%253e%253c%2500rhLvZ%253e |

dataset in these days is run by 5 times. The average run-time of these models is recorded in the unit of seconds and illustrated in the table. As showed in the table, the common-feature model performs better providing that the number of log records below 600,000 but the model is quite slower when the log records above 1 million. Despite its simplicity in computing feature, the common-feature model cannot keep pace with TF-IDF model when data size increases.

## 4.5 Discussions

Applying rule generation techniques to anomaly detection helps administrators easily visualize how the detection system works. Machine learning techniques using the decision tree algorithm allow the development of anomaly detection rules quickly and efficiently. This technique is also one of the common and typical for detecting anomalies based on the generation of anomalous behavioral classification tree. The experiment result showed that using TD-IDF features to represent user behavior from access log data achieves good results compared to other representation.

The advantage of a decision-based technique is that the training speed is fast but the effectiveness of the detection system depends on the quality of the dataset used to build the analytical model. The report proposes a way to build a dataset that meets the need for detecting anomaly and monitoring user access based on the ZAP security tool. This dataset is stored in semi-structured files including anomaly (anomaly.csv) and normal (normal.csv) samples, and blacklists (blacklist.csv). This greatly supports administration in analysis and monitoring of web services with limited resources. The simple structure through semi-structured files allows the administrator to append malicious or normal accesses. In other words, administrators or operators of web services can maintain a library of user access behaviors that appropriately accommodates to

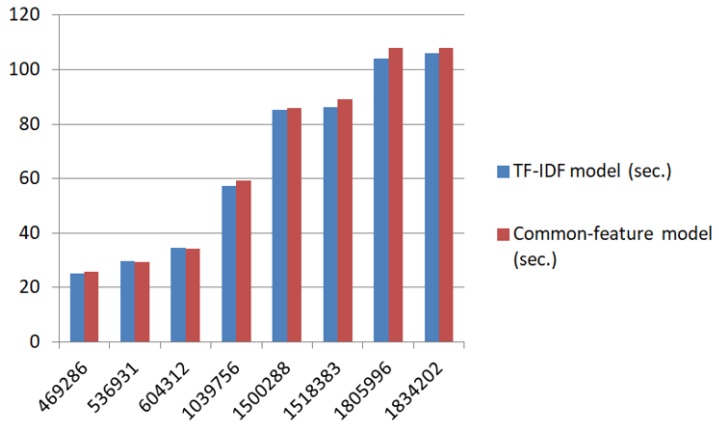


Figure 6: Comparison of the run-time of the two models (TF-IDF and common-feature) during testing phase (detecting anomaly)

their own needs.

The anomaly detection system proposed to use a group of decision tree algorithms, namely random forests, based on an assessment of the training rate, performance and abnormal performance. On the other hand, the use of other algorithms such as SVM machine learning vector or deep learning techniques, can also improve the detection performance of the abnormal classification model obtained. However, these techniques are quite limited at the complexity of deployment and training time, and may require special hardware and software (especially for deep learning techniques).

The proposed system has been performing an experimental analysis of quite a large amount of data up to millions of user accesses. The Python environment and NoSQL MongoDB database combine quite well when handling such volumes. However, the proposed system is not really geared towards handling big data like the Apache Spark platform. Even though, large data processing platforms often provide toolkits connected to the Python environment due to the popularity of this environment. It is possible to integrate and extend the proposed system with large data processing platforms like Apache Spark.

# 5 Conclusion

With the increasing popularity of web services, the issue of administration and monitoring user behaviors becomes even more urgent to ensure the quality of service as well as the security of the web services. Anomaly detection in web services can range from detecting misuse of users to malicious purposes which

degrade the quality of website service to commit fraudulent behaviors.

The paper explores how to detect unusual accesses from log data on a web server based on automatic rules generation by applying random forest algorithms. User access to the services is represented by TF-IDF feature thanks to its detecting performance. In addition, the report presented the method to build and maintain a dataset for the development of an extraordinary classification model based on the ZAP security tool. Administrators can easily maintain and update datasets according to their own management and supervision needs. Testing on the proposed anomaly detection system shows that the system works relatively well, reaches 95% accurate detection and is capable of processing and monitoring the volume of query data up to millions of Records of user queries.

In the future, anomaly detection systems could be further investigated to incorporate more advanced machine learning algorithms to enhance the anomaly detection performance. On the other hand, the analysis of anomalous access behavior can be more detailed such as XSS, SQL or hijacking instead of normal and abnormal as currently. Integration with large data processing platforms is also a practical task to meet the needs of administration and monitoring with large-scale web services.

# References

[1] De Stefano, Claudio, Carlo Sansone, and Mario Vento. "To reject or not to reject: that is the question-an answer in case of neural classifiers." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.1 (2000): 84-94.

[2] Barbara, Daniel, Ningning Wu, and Sushil Jajodia. "Detecting novel network intrusions using bayes estimators." *Proceedings of the 2001 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2001.

[3] Fan, Wei, et al. "Using artificial anomalies to detect unknown and known network intrusions." *Knowledge and Information Systems* 6.5 (2004): 507-527.

[4] Helmer, Guy G., et al. "Intelligent agents for intrusion detection." *1998 IEEE Information Technology Conference, Information Environment for the Future (Cat. No. 98EX228)*. IEEE, 1998.

[5] Lee, Wenke, Salvatore J. Stolfo, and Philip K. Chan. "Learning patterns from unix process execution traces for intrusion detection." *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*. 1997.

[6] Salvador, Stan, Philip Chan, and John Brodie. "Learning States and Rules for Time Series Anomaly Detection." *FLAIRS conference*. 2004.

[7] Teng, Henry S., Kaihu Chen, and Stephen C. Lu. "Security audit trail analysis using inductively generated predictive rules." *Sixth Conference on Artificial Intelligence for Applications*. IEEE, 1990.

[8] Agrawal, Rakesh, and Ramakrishnan Srikant. "Mining sequential patterns." *icde*. Vol. 95. 1995.

[9] Mahoney, Matthew V., and Philip K. Chan. Learning rules for anomaly detection of hostile network traffic. 2003.

[10] Chan, Philip K., Matthew V. Mahoney, and Muhammad H. Arshad. A machine learning approach to anomaly detection. 2003.

[11] Tandon, Gaurav, and Philip K. Chan. "Weighting versus pruning in rule validation for detecting network and host anomalies." *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007.

[12] Chan, Gaik-Yee, Chien-Sing Lee, and Swee-Huay Heng. "Discovering fuzzy association rule patterns and increasing sensitivity analysis of XML-related attacks." *Journal of Network and Computer Applications* 36.2 (2013): 829-842.

[13] Ezeife, Christie I., Jingyu Dong, and Akshai K. Aggarwal. "SensorWebIDS: a web mining intrusion detection system." *International Journal of web Information Systems 4.1* (2008): 97-120.

[14] Breiman, Leo. "Random Forests." *Machine learning* 45.1 (2001): 5-32.

[15] Nguyen, Hai Thanh, et al. "Application of the generic feature selection measure in detection of web attacks." *Computational Intelligence in Security for Information Systems*. Springer, Berlin, Heidelberg, 2011. 25-32.

[16] Christopher, D. Manning, Raghavan Prabhakar, and Schtze Hinrich. "Introduction to information retrieval." *An Introduction To Information Retrieval* 151.177 (2008): 5.

[17] Gimnez, Carmen Torrano, Alejandro Prez Villegas, and Gonzalo lvarez Maran. "HTTP dataset CSIC 2010." *Information Security Institute of CSIC* (Spanish Research National Council) (2010).

[18] Bennetts, Simon. "Owasp zed attack proxy." *AppSec USA* (2013).