International Journal of Research and Review
www.ijrrjournal.com E-ISSN: 2349-9788; P-ISSN: 2454-2237

Original Research Article

Detection of Anomalous users in Web Applications using Fuzzy Logic

Rasoul Jourmand[1], Seyed Enayatallah Alavi[2]

[1]MSc Student, Institute for Higher Education ACECR Khuzestan, Ahwaz, Iran,
[2]Assistant Professor, Shahid Chamran University of Ahwaz, Ahwaz, Iran.

Corresponding Author: Rasoul Jourmand

Received: 30/06/2015 *Revised: 02/07/2015* *Accepted: 11/07/2015*

ABSTRACT

In this paper, we present an approach based on XML files, which can be implemented independently from the application layer in a proxy server, to perform related operations to detect abnormalities in the web applications without the need for initial training. Normal behavior of web applications is automatically and accurately based on database structure and web pages defined in XML files. In this approach all requests sent by user browser are received and analyzed, so if receiving requests weren't match with the configured security model, they will never reach web server. In addition, to detect anomalous users fuzzy logic has taken interest. This approach has been implemented in Medical Sciences Research Automation system (Syat) of Mazandaran, Iran. The results of this evaluation shows that using fuzzy logic to discover anomalous users comes along with flexibility, increased accuracy and reducing false detects.

Key Words: Security - Anomaly Detection - Web Application - Fuzzy logic.

INTRODUCTION

The World Wide Web has become a prevalent platform for information dissemination and service delivery. In addition to publishing information, web applications are increasingly deployed as portals to support a wide range of business processes, content management and value-added services. [1]

Most security concerns are now related with the application level. This has one simple explanation, web applications are accessible through browsers, and can be accessed by everyone with Internet. [2] This has the inherent possibility that good or badly intentioned people can take advantage of this and perform malicious actions. The number of attacks documented by some entities easily confirms this. The National Institute of Standards and Technology (NIST) hold a National Vulnerability Database (NVD), which has over 40000 vulnerabilities, identified in the application level as of March 13, 2010. [3] This is also confirmed by the Gartner Group, which estimates that 70% of the attacks to a company's web application come from the application level. [4]

Accordingly, Detection of anomalous behavior of users in the Internet-based business web applications and portals is very important in maintaining security. Such users' behavior can have a significant impact on the web application. [5] In order to protect web applications, proper security solutions should be taken. So a flexible and

a secure model are required against such attacks.

Different researches accomplished in this field. For instance, the system presented in [6] is a kind of firewall which analyzes HTTP requests. This system requires a precise picture of web application's natural behavior before detection process. This picture is created in learning phase. Advantages of this approach are: scalability, ability to detect different types of attacks, optimized usage of resources. Beside the following advantages the following flaws are exist too: defining regular traffic in large and complicated web applications isn't simple and detecting the issue which has caused the alert isn't possible.

The presented approach by Kirchner in, [7] is providing a comprehensive reverse proxy framework for the detection of anomalies in HTTP traffic. It enables an automated adaption to application specific usage patterns and offers an incremental learning feature. Instead of using generic signatures of common attacks, this framework uses instance based learning and extracts usage patterns for each URI of a web application.

Researches in, [8] analysis of HTTP logs for the detection of network intrusions. First, a training set of HTTP requests which does not contain any attacks is analyzed. When all relevant information has been extracted from the logs, several clustering and anomaly detection algorithms are employed to describe the model of normal user's behavior. To describe the model of legitimate user behavior, several data mining techniques are employed. These techniques are based on unsupervised machine learning and, therefore, allow one to build the model without a priori knowledge about the structure of the data in the training set.

Authors in [9] presented a data-base firewall to prevent from attacks against back-end data-base of web applications. The firewall listens SQL query requests from the client as well as analyzes them, and then if they are safe, will call the original SQL server to execute the queries, else will block the queries and return an empty result to the client. One of the required levels in this approach is learning phase. In learning mode, the firewall learns normal data-base accessing features of the legitimate users. It should be noted that in order to obtain proper results, learning phase should be done completely and with accuracy. [10]

Authors of [1] presented an approach which based on hidden markov Model (HMM). This approach is capable to recognizing anomalous user's activity in Workflow-driven Web Applications. It uses markov chain to predict users' behavior. The advantages of this approach are: independency from application code and it is suitable for web applications using complicated supportive politics. But disadvantages of this approach are: requirement to learning phase and high rate of false alarms.

According to existing challenges and flaws in prior approaches, the approach presented in this research is not only able to remove limits from available approaches but it also with high accuracy can detects abnormal users who may harm the web application. If an abnormal user was detected by this system, that user's further activity will be prevented.

MATERIALS AND METHODS

Purposed Framework:

The system presented in this paper is a simple Web Application Firewall which analyzes HTTP requests sent by a client browser trying to get access to a web server. The analysis takes place exclusively at the application layer. The system follows the anomaly-based approach, detecting known and unknown web attacks.

In our architecture, the system operates as a proxy located between the client and the web server. Likewise, the system might be embedded as a module within the server. However, the first approach enjoys the advantage of being independent of the web platform.

This proxy analyzes all the traffic sent by the client. The input of the detection process consists of a collection of HTTP requests$\{r_1, r_2, \dots, r_n\}$. The output is number of anomalies a_i for each input request r_i, which indicates whether the request is normal or anomalous.

Prior to the detection process, the system needs a precise picture of what the normal behavior is in a specific web application. For this purpose, our system relies on an XML file which contains a thorough description of the web application's normal behavior. Once a request is received, the system compares it with the normal behavior model. If the difference exceeds the given thresholds, then the request is flagged as an attack and an alert is launched. In this approach, the XML files are created automatically by the system. Figure 1 shows a general view of anomaly detection.

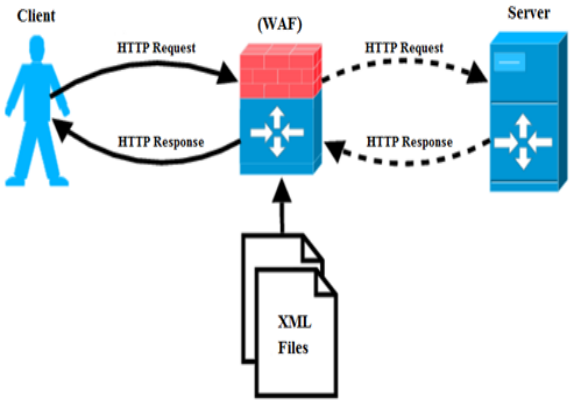

Figure 1. General view of anomaly detection

Afterwards, fuzzy logic has been used to detect anomalous users, which brings increased accuracy and flexibility. So based on a series of invalid requests, it can determine the level of harmful and accordingly fine can be applied based on harmfulness.

Implementation of The proposed Framework:

The framework Presented in this approach was developed by Microsoft .Net software package. Since this approach is an independent framework from the application, so there is no persuasion on matching the host application technology with proposed framework.

The current approach focuses on sent requests to the application. So user entries, HTTP Headers, HTTP Verbs and other existing sections in a request should be inspected completely. To begin, the acceptable entry for each field in the web application should be defined precisely. For example, for a field like "Phone Number", acceptable characters, string length and etc. are defined clearly. To define corresponding fields "Regular Expressions" (or in short term "Regex") can be used. Since most programming languages (PHP, Java, JavaScript and etc.) are using Regular Expressions as uniform standard, so we decided to use this feature in the following approach.

Accordingly, we will define all available fields in database tables inside a table named "Field Regex". In the future, this table can be used for different projects as a module. To simplify different operations over this table (Add, Update, Delete and read records), CRUD forms can be designed to make it easy in the future to Add, Edit and Delete regular expressions for new fields.

In this way, defining database table fields might take time, but it should be noted that only fields with specific entry will be defined. In addition, to define rules as regular expressions different software in this filed are released which can be useful (Including Reg Exr v2.0, which is available

for free on the Web [11]). Moreover, developers must use regular expressions to check incoming entries during development of critical applications.

In the next level, different parts of incoming requests for each web page should be defined. These parts include HTTP Verbs, HTTP Headers and Pages Location. Specifying each one of these parts for many web forms can take a long time. For this reason, we can do this as well as other parts automatically by coding and obtaining needed information from web pages structure. After defining this structure, creating XML files comes next. Therefore, fields related to each table are called and according to their name and information, related XML file to that table will be generated. To this end, the field name is sent to Field Regex table and the expression related to that field is acquired. This expression will be placed into XML file related to that table.

There are also situations in which the field name definition is not present in the table of rules ("FieldRegex" table). In this situation, information related to that field will be used. For example, if "Name" field doesn't exist in "FieldRegex" table, the field type (e.g. String, Numeric, Decimal and etc.), Character Length and is "Required" or not are used and regular expression will be generated afterwards.

Normal Behavior Description:

The XML file contains rules regarding to the correctness of HTTP verbs, HTTP headers, accessed resources (files), arguments, and values for the arguments. This file contains three main nodes:

- Verbs. The verbs node simply specifies the list of allowed HTTP verbs.
- Headers. The headers node specifies a list of some HTTP headers and their allowed values.
- Directories. The directories node has a tree-like structure, in close correspondence to the web application's directory structure.

1. Each directory in the web application space is represented in the XML file by a directory node, allowing nesting of directories within directories. The attribute name defines these nodes.
2. Each file in the web application space is represented by a file node within a directory node and is defined by its attribute name.
3. Input arguments are represented by argument nodes within the corresponding file node. Each argument is defined by its name and a Boolean value required Field indicating whether the request should be rejected if the argument is missing.
4. Legal values for arguments should meet some statistical rules, which are represented by a stats node within the corresponding argument node. These descriptions which are presented as regular expression give a description of the expected values. An example of XML configuration file is shown in Fig. 2.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<configuration>
  <verbs>
    <verb>GET</verb>
    <verb>POST</verb>
  </verbs>
  <headers>
    <rule name="Accept-Charset" value="ISO-8859-1"/>
  </headers>
  <directory name="admin">
    <file name="addUser.aspx">
      <argument name="Name" requiredField="true">
        <stats discriptin="/^[a-z0-9]$/i*\d{10}$" />
      </argument>
      <argument name="Email" requiredField="false">
        <stats discriptin="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*" />
      </argument>
      <argument name="Phone" requiredField="true">
        <stats discriptin="/^(((\+|00)98)|0)?9[123]\d{8}$/" />
      </argument>
      <argument name="CodeMeli" requiredField="true">
        <stats discriptin="^\d{10}$" />
      </argument>
    …
  </directory>
</configuration>
```

Figure 2. Example of XML configuration file

Detection Process:

In the detection process, our system follows an approach of the form "deny everything unless explicitly allowed", also known as positive security model.

The detection process takes place in the proxy. It consists of several steps, each constituting a different line of defense, in which the different parts of the request are checked with the aid of the XML file. If an incoming request fails to pass one of these lines of defense, an attack is assumed: a customizable error page is returned to the user and the request is logged for further inspection. It is important to stress that these requests will never reach the web server when operating in prevention mode.

The detection process is composed of the following steps:

1. Verb check. The verb must be present in the XML file, otherwise the request is rejected. For example, in the applications in which only GET, POST and HEAD are required to work correctly, the XML file could be configured accordingly, thus rejecting requests that use any other verb.
2. Headers check. If the header appears in the XML file, its value must be included too. Different values will not be accepted, thus preventing attacks embedded in these elements.
3. Resource test. The system checks whether the requested resource is valid. For this purpose, the XML configuration file contains a complete list of all files that are allowed to be served. If the requested resource is not present in the list, a web attack is assumed.
4. Arguments test. If the request has any argument, the following aspects are checked:

a) It is checked that all arguments are allowed for the resource. If the request includes arguments not listed in the XML file for the corresponding resource, a manipulation of the arguments is assumed and thus the request is rejected.

b) It is confirmed that all mandatory arguments are present in the request. If any mandatory argument (required Field="true") is not present in the request, it is rejected.

c) Argument values are checked. An incoming request will be allowed if all parameter values are identified as normal. Argument values are decoded before being checked. For the argument value test, the statistical properties of the corresponding argument are used. If any property of the argument is outside the corresponding interval or contains any forbidden special character, the request is rejected.

These steps allow the detection of both static attacks, which request resources that do not belong to the application, and dynamic attacks, which manipulate the arguments of the request.

Using Fuzzy Logic:

To increase detection accuracy and reducing false detections rate fuzzy logic has been applied. Therefore, it gives us more confidence to detect abnormal users and preventing further activities of malicious users in the web application. Considering using fuzzy logic, it converts our system to a mixed security model. While before using fuzzy logic our system was known as a positive security model. Consequently, in this approach we can take the advantages of both negative and positive models.

To this end, several parts soft he web application will be examined. One of these parts is the obtained output from anomaly detection. The value of this output based on the user's anomalous activity and repeats,

has a range between 0 to greater than 1 for each user. The formula to calculate this value is shown in equation 1:

1. $\boldsymbol{a + (C \times 2) = Anomaly\_Range}$

In this equation, "a" stands for the numbers of anomaly in a request and "c" equals to all abnormalities occurred by the same user in past 24 hours. "Anomaly Range" variable in the equation contains the result. For each anomaly in the incoming requests, 0.1 will be considered.

Another important part in most web applications is Login ability. This ability is used to keep user's information safe and to prevent unauthorized access. [12,13] We want to decrease error rate of anomaly detection and refuse fast judgment. For this purpose we will use equation 2:

2. $\boldsymbol{F + (C \times 2) = Anomaly\_Range}$

In this equation, "F" stands for the number of incorrect characters that entered in the Login form, "C" is the number of Current user retries to log in to application in past 42 hours and "Anomaly Range" is the result. The output is always between 0 to greater than 2.

The last part that we're going to monitor is IP Address. IP address is always the most important identifier to track users. So observing IP address is an important task. [14,15] To obtain the rate of IP address changes for each user, we will use an online small library. This library according to the changing Internet service provider, geographic location and etc. current IP address compared with the previous IP address, gives an output number between 0 to 12.

Now, according to defined membership functions, necessary rules to detect abnormal users can be deduced. In the following, there are a set of rules defined that their results shows the user is normal. So, in other status the user considered as abnormal. The results of these rules according to the level of required security in the application can be different.

- IF Anomalous_Request = low And Anomalies_Login = low And Anomalies_IP_address = low Then User = Non-Anomalous
- IF Anomalous_Request = low And Anomalies_Login = low And Anomalies_IP_address = Moderate Then User = Non-Anomalous
- IF Anomalous_Request = low And Anomalies_Login = Moderate And Anomalies_IP_address = low Then User = Non-Anomalous
- IF Anomalous_Request = Moderate and Anomalies_Login = low And Anomalies_IP_address = low Then User = Non-Anomalous

RESULTS AND DISCUSSION

Our evaluation study is performed over a system, Syat, [16] which is a Medical Sciences Research Automation system that has been in operation at Mazandaran, IRAN University Medical Sciences for over two years. Researchers can submit their research plans and other users such as judges, experts and etc. are responsible to handle submitted researches. This system developed with ASP.NET and C# Language. The database used for this system is developed in SQL Server Management Studio 2008 R2.

To evaluate the effectiveness of the proposed approach, the assessment criteria of true positives (number of legitimate requests that have been detected correctly), true negative (number of abnormal requests that have been detected correctly), false positives (number of legitimate requests that have been detected as abnormal) and false negative (number of abnormal requests that have been detected as legitimate) will be used. But it should be noted that "false negative" in web applications is too important because if anomalous request

reaches the web server it can cause irreparable damages.

In this assessment, the model proposed by Xiaowei et al. [1] and also the proposed model by Torrano-Giménezet al. [6] were implemented and compared with the current model. The model presented in [1] was based on HMM and the model in [6] is an approach for detect anomaly in web applications. To this end, each of these approaches was tested in up to 3 thousand requests. In this collection, 2128 requests were normal (without any anomaly) and 827 of them were abnormal. The results obtained from this evaluation are described in Table 1 and it is shown as a graph in figure 3.

Table 1.Results obtained from each approach evaluation

	True Positive	True Negative	False Positive	False Negative
Xiaowei et al. [1]	1832	524	296	348
Torrano-Giménez et al. [6]	1994	638	134	234
Current Approach	2121	872	7	0

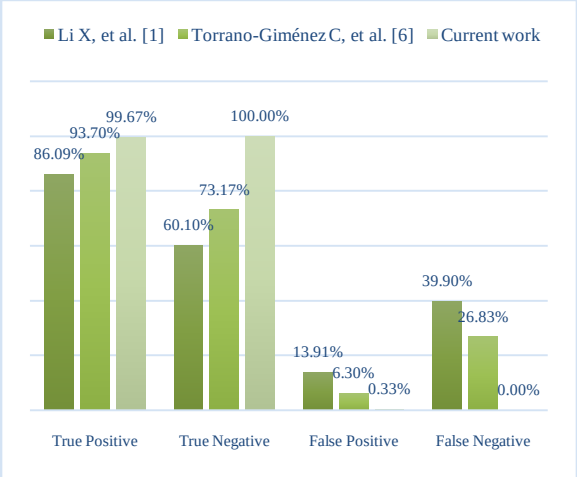

Figure 3.Results obtained from each approach evaluation

But the main objective of this research is to detect anomalous users. For this purpose, we have compared obtained results of 3 months activity from this approach before and after using fuzzy logic in Syat system (shown in figure 4).

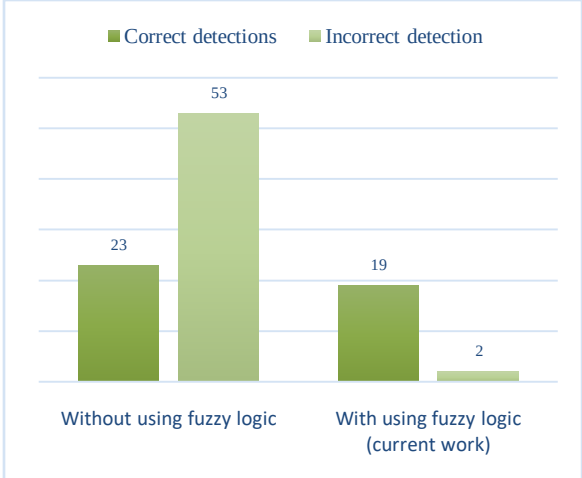

Figure 4.Obtained results before and after using fuzzy logic

CONCLUSION

Presented approach in this research can analyze incoming requests from end-users before reaching web server and preventing those requests if they were containing any attacks. This approach is XML based and it can be implemented independently in a proxy server. So, independency from Application Layer is an advantage of this approach. Accordingly, it can be easily integrated with other developed applications.

Another advantage of this approach is automatic training. While most anomaly detection approaches in the web applications require this training phase. This phase is the foundation of these approaches and it should be done manually before releasing application in operating phase. But in this approach, training process is without human intervention and it will be done based on database table fields and web pages structure. Hence it can help to the rapid deployment of applications.

On the other hand, implementing existing anomaly detection approaches in large applications is a very difficult task. Because each part of the application should be trained with thousands of valid requests. In addition, applications will face many changes during their lifetime. So, by

following any of these approaches and any changes, the long lasting and difficult level of training needs to be restarted. It should be noted that training phase in these approaches should be done with high accuracy, because in case of any existing weakness in this phase, the system can't discover any attacks.

REFERENCES

1. Xiaowei L, Yuan X, Bradley M. Detecting Anomalous User Behaviors in Workflow-Driven Web Applications. Proceedings of the 31st Symposium on Reliable Distributed Systems; 2012 Oct 8-11; Irvine, California. IEEE; 2012. p. 1-10.
2. Viswanathan RP, Al-Nashif Y, Hariri S. Application attack detection system (AADS): An anomaly based behavior analysis approach. 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA); 2011 Dec 27-30; Sharm El-Sheikh, Egypt. IEEE; 2011. p. 406-411.
3. Wang C, Viswanathan K, Choudur L, et al. Statistical techniques for online anomaly detection in data centers. IFIP/IEEE International Symposium on Integrated Network Management (IM); 2011 May 23-27; Dublin, Ireland. IEEE; 2011. p. 385-392.
4. Antunes N, Vieira M. Defending against web application vulnerabilities. Journal of Computer. 2012; 45(2): 66-72.
5. Threepak T, Watcharapupong A. Web attack detection using entropy-based analysis. International Conference on Information Networking (ICOIN); 2014 Feb 10-12; Phuket, Thailand. IEEE; 2011. p. 244-247.
6. Torrano-Giménez C, Perez-Villegas A, ÁlvarezMarañón, G. An Anomaly-Based Approach for Intrusion Detection in Web Traffic. Journal of Information Assurance and Security (JIAS). 2010; 5(4):446-454.
7. Kirchner M. A framework for detecting anomalies in HTTP traffic using instance-based learning and k-nearest neighbor classification. 2nd International Workshop on Security and Communication Networks (IWSCN); 2010May 26-28; Karlstad, Sweden. IEEE; 2010. p. 1-8.
8. Zolotukhin M, Hamalainen T, Kokkonen T, et al. Analysis of HTTP Requests for Anomaly Detection of Web Attacks. 12th International Conference on Dependable, Autonomic and Secure Computing (DASC); 2014 Aug 24-27; Dalian, China. IEEE; 2014. p. 406-411.
9. Sun J, Chen H, Niu C. A New Database Firewall Based on Anomaly Detection. International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT); 2010 Dec 8-11; Wuhan, China. IEEE; 2010. p. 399-404.
10. Ruzhi X, Liwu D. A learning-based anomaly detection model of SQL attacks. 2010 IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS); 2010 June 25-27; Beijing, China. IEEE; 2010. p. 639-642.
11. "RegExr v2.0." [Online]. Available: http://www.regexr.com/
12. Kumar R. Mitigating the authentication vulnerabilities in Web applications through security requirements. World Congress on Information and Communication Technologies (WICT); 2011 Dec 11-14; Mumbai, India. IEEE; 2011. p. 1294-1298.
13. Salini P, S Kanmani. Security Requirements Engineering Process for Web Applications. Procedia Engineering. 2012; 38(0): 2799-2807.
14. Atashzar H, Torkaman A, Bahrololum M, et al. A survey on web application vulnerabilities and countermeasures. 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT); 2011 Nov 29-Dec 1; Seogwipo, South Korea. IEEE; 2011. p. 647-652.
15. Teodoro N, Serrao C. Web application security: Improving critical web-based

applications quality through in-depth security analysis. International Conference on Information Society (i-Society); 2011 June 27-29; London, England. IEEE; 2011. p. 457-462.

16. "Syat: Medical Sciences Research Automation system." [Online]. Available: http://syat.mazums.ac.ir/

How to cite this article: Jourmand R, Alavi SE. Detection of anomalous users in web applications using fuzzy logic. Int J Res Rev. 2015; 2(7):406-414.

****