International Journal of Advanced Engineering & Innovative Technology (IJAEIT)
ISSN: 2348 7208
ISRA IMPACT FACTOR: 0.42

SQL-I Intrusion Protection against SQL Injection Attacks using RANDOM4 and Fuzzy Clustering-Artificial Neural Network

Swarangi Todankar, Bhagyashree Vanarse, Samrudhi Walunj, Pallavi Lokhande

Abstract—The Web Applications which we are using in present times contain huge amount of information. But the implementation of these applications is not very much secured. They are vulnerable to a variety of web based threats. One of these threats is SQL injection attacks. In these kind of attacks the attacker inserts a set of malicious string contents. By using these attacks the attackers tries to gain access to the information's stored in the databases of these applications. SQL Injection is the act of passing SQL code into web applications, such attacks target interactive web applications that employ database services. By employing SQL Injection Attacks, attackers can leak confidential information and even corrupt the database. To prevent the SQL Injection Attacks we are using Random4 algorithm and FC-ANN algorithm. The random4 algorithm is based on randomization and is used to convert the input into a cipher text incorporating the concept of cryptographic .All known attacks is prevented by random4. This is the drawback of Reverse Proxy Algorithm. To overcome this problem we are implementing Fuzzy clustering- Artificial Neural Network. Through fuzzy clustering technique, the heterogeneous training set of attack is divided to several homogeneous subsets. Thus complexity of each sub training set is reduced and consequently the detection performance is increase.

Index Terms— Randomization; SQL injection; Vulnerability; web applications; Intrusion detection systems; Artificial Neural Networks; Fuzzy Clustering; decision precision and detection stability.

1. INTRODUCTION

According to the report by the White Hat on web security vulnerabilities 2011, it shows that nearly 14-15 % of web application attacks account for SQL Injection [1]. In this paper we take up SQL Injection, a critical web security vulnerability. SQLIA is a type of code-injection attack [2]. It is caused mainly due to improper validation of user input. Solutions addressed to prevent SQL Injection Attack include existing defensive coding practices alongside encryption algorithms based on randomization & FC-ANN. The main intent to use SQL injection attack include illegal access to a database, extracting information from the database, modifying the existing database, escalation of privileges of the user or to malfunction an application. Ultimately SQLIA involves unauthorized access to a database exploiting the vulnerable parameters of a web application.

Swarangi Todankar, Bhagyashree Vanarse, Samrudhi Walunj, Pallavi Lokhande, Reserach students of Padmabhooshan Vasantdada Patil Institute Of Technology, Bavdhan. *Pune 411 021*

A novel idea to detect and prevent SQLIA, an application specific encryption algorithm based on randomization is proposed and its effectiveness is measured.

Intrusion detection attempts to detect computer attacks by examining various data records observed in processes on the network, It is one of the important ways to solve network security problems. Detection precision and detection stability are two key indicators to evaluate intrusion detection systems (IDS). For the above two aspects, the main reason is that the distribution of different types of attacks is imbalanced. For low-frequent attacks, the size is too small compared to high-frequent attacks. It makes ANN not easy to learn the characters of these attacks and therefore detection precision is much lower. In practice, low-frequent attacks do not mean they are unimportant. Instead, serious consequence will be caused if these attacks succeeded. For example, if the U2R attacks succeeded, the attacker can get the authority of root user and do everything he likes to the targeted computer systems or network device. To solve the above two problems, we propose a novel approach for ANN-based IDS, FC-ANN, to enhance the detection precision for low-frequent attacks and detection stability.

2. SQL INJECTION

In order to locate where SQLIA vulnerability occurs, we first discuss about the 3-tier logical view architecture of web applications [1].

2.1.3-tier Architecture of web application

1) User interface tier: This layer forms the front end of the web application. It interacts with the other layers based on the inputs provided by the user.

2) Business logic tier: The user request and its processing are done here. It involves the server side programming logic. Forms the intermediate layer between the user interface tier and the database tier.

3) Database tier: It involves the database server. It is useful in storage and retrieval of data.

2.2. Basic principle in SQL injection

SQL injection attack is a web security attack by using SQL statements exploiting the poorly designed input elements of a web form. This compromises the confidentiality and integrity of users' sensitive data. SQLIA takes place between the user interface layer and the business-logic layer. To understand the essence of SQL injection let us see the following example.

SELECT * from tablename WHERE user=' ' and password= ' ';

A sample SQL statement containing two input parameters is considered .Instead of typing the actual username and password, if a hacker attempts illegally to access the database by inputting SQL statements, it is said to be a SQLi attempt. For example if the hacker inputs, ' OR'1'='1' - -, the statement becomes,

SELECT * from tablename WHERE user='' OR '1'='1'- - and password= ' ' ;

Here the user gets unauthorized access to the system because 1=1 is true always and - - indicates the statements following it are comments. Therefore, if the inputs by the user were not properly sanitized, it might lead to a critical web security attack. This describes the basic principle involved in SQL injection.

2.3. Attack Types:

This section briefly describes about the types of SQLi attacks for which we propose solutions in the next section.

2.3.1. Using tautology: The example quoted in the section II.B to explain SQL injection describes the attack using tautology.

2.3.2. Using illegal/incorrect queries:

By providing incorrect inputs, the database might return some important information regarding the table and fields used in the database. Using successive requests like these, the security can be compromised. In the following example instead of avalid username (xyz), an incorrect input (xyz') is provided and an error message is returned giving clues about the database making it vulnerable to intrusion.

Error: SELECT username, password from student WHERE username = xyz'

Here field names such as username and password from table student are exposed.

2.3.3. Using Piggy-Backed Queries: Previously mentioned attack types try to gain unauthorized access to the application or fetch details about the database without any additional queries added to the input query. If additional queries are piggy-backed in the input area using special characters such as " ", " – " or " ; " hackers can modify or delete the database. An example to this is:

SELECT * from User WHERE id=123;drop table User;

Here ";" acts as a delimiter and additional an drop statement may be executed and deletes the table.

2.3.4. Blind injection: In case of incorrect queries, programmers try to hide database information as a measure to protect the application from attacks using illegal/incorrect queries. Now, the hacker does not get any clues about the database. The option hence used to compromise security is,by querying the database with a lot of true/false queries and checking its result. Based on the response of the application to the queries, the hackers attempt illegal access. This type of attack is most prominently used. It includes timing attacks as one of the technique to identify the behavior of the application.

2.3.5.DDOS attack :

A data center which provides internet service may suffer from many security risks including Distributed Denial of Service (DDOS) attack. We implemented the proposed system and evaluated the performance to show that our system works efficiently to mitigate the DDos traffic from the Internet.

3. LITERATURE SURVEY

As mentioned earlier, various works have been done to protect web applications from SQL injection attack. This section will brief on SQL Injection detection technique, and also touches FC-ANN and Reverse Proxy.

3.1. An Intelligent Categorization Engine [7]: The web content categorization engine performs analysis on the nature of the web pages, and categorizes web pages into "neutral" and "objectionable". Web pages categorized as objectionable are blocked from reaching the users.

3.2. Positive Tainting and Syntax-Aware Evaluation [8]: This approach taking advantage of the characteristics of SQLIAs and Web applications. First, positive tainting, that is, the identification and marking of trusted, instead of entrusted, data. Second, our approach performs accurate and efficient taint propagation by precisely tracking trust markings at the character level. Third, it performs syntax-aware evaluation of query strings before they are sent to the database and blocks all queries whose non literal parts (that is, SQL keywords and operators) contain one or more characters without trust markings. Finally, our approach has minimal deployment requirements.

3.3. Log Visualization[9]: From the log, a web security administrator can really understand what is happening to the network, uncover hidden pattern, thus identify and respond to the attacks accordingly Through visualization, any complex protocol or other security-related ideas can be well-depict.

4. CURRENT SYSTEM

Basically in this system SQL injection attacks are prevented. Prevention and detection of attack is done by IDS probability of attack using reverse proxy and FC-ANN.IDS send information to the admin after verification. If attack is detected by IDS then admin inform client about attack otherwise give access to server for transaction

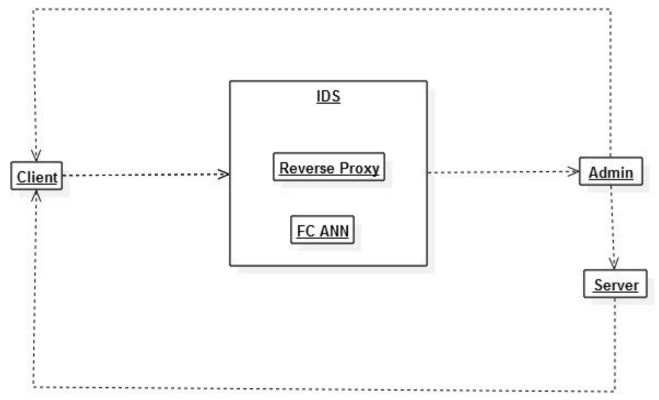


Fig 1: System Architecture

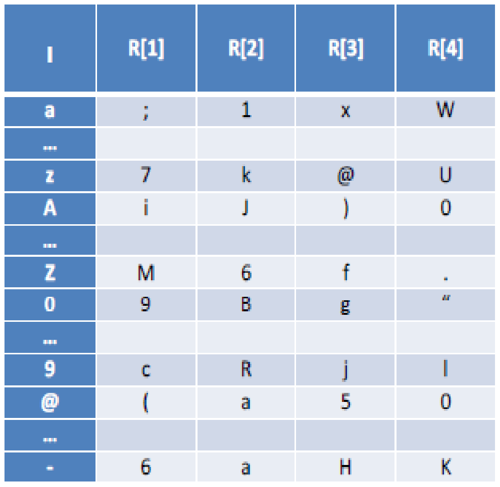
Reverse Proxy is implemented with the help of random4 encryption algorithm to prevent SQLi these to generate the cipher text based on randomized table. Further this attacks are checked for unknown attacks which are prevented by FC-ANN.

4.1. Random4 Algorithm and Framework

The random4 algorithm is based on randomization and is used to convert the input into a cipher text incorporating the concept of cryptographic salt. This algorithm forms the basis of the proposed approach. Any input in web forms will contain numbers, uppercase, lowercase or special characters. Keeping this in mind the input from user is encrypted based on randomization.

ALGORITHM RANDOM4:

```
Input    : input string ip[]
Output   : Encrypted string en[]
N        : Length of ip[]
R[]      :Random values of character
For i=1 to N
        if (ip[i+1]=null || ip[i+1]=lowercase)
                then en[i]=R[1]
        End { if }
        else if(ip[i+1]=uppercase)
                then en[i]=R[2]
        End {else if }
        else if(ip[i+1]=number)
                then en[i]=R[3]
        End {else if }
        else if(ip[i+1]=spl.char)
                then en[i]= R[4]
        End {else if }
End { For }
return en[]
```



I	R[1]	R[2]	R[3]	R[4]
a	;	1	x	W
...				
z	7	k	@	U
A	i	J	)	0
...				
Z	M	6	f	.
0	9	B	g	"
...				
9	c	R	j	l
@	(	a	5	0
...				
-	6	a	H	K

Fig 2: Lookup table for Random4 algorithm

Framework for RANDOM4

A framework to build the tool generating the encrypted text based on Random4 is shown in [Fig6]. A normal text is given as input . Each input character is mapped to one of its four values as in the lookup table. The key values of each character are concatenated to form the cipher text.

This can then be stored in database and used by programmers in server side programming to prevent illegal access through SQL injection. A sample screen shot of the output generated by this tool.

4.2. FC-ANN: In this section, we elaborate our new approach, FC-ANN. We firstly present the whole framework of the new approach. Then we discuss the three main modules, i.e., fuzzy clustering module, ANN module, and fuzzy aggregation module.

The training phase includes the following three major stages:

Stage I: For an arbitrary data set DS, it is firstly divided into training set TR and testing set TS. Then the different training subsets TR1; TR2; . . . ; TRk are created from TR with fuzzy clustering module.

Stage II: For each training subset TRi is training by the specific learning algorithm to formulate k different base ANN models.

Stage III: In order to reduce the error for every ANNi, we simulate the ANNi using the whole training set TR and get the results. Then we use the membership grades, which were generated by fuzzy clustering module, to combine the results. Subsequently, we train another new ANN

Using the combined results. In the testing phase, we directly input the testing set data intothe k different ANNi and get outputs. Based on these outputs, thefinal results can then be achieved by the last fuzzy aggregationmodule.

5. CONCLUSION

This paper highlights SQLIAs and how the combination of a reverse proxy with FC-ANN can help organizations to detect and block SQLIA in efficient manner. This system has the facility to detect unknown attacks that are newly evolved with the help of ANN algorithm apart from known attacks. But this system does not know full details of any low or high frequent attacks therefore it checks attacks irrespective of frequency. Therefore we conclude that this system is updated with new attack prevention for web application

6. REFERENCES

[1] https://www.whitehatsec.com/resource/stats.html

[2] W. G. Halfond and A. Orso. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proceedings of the IEEE and ACM International Conference on Automated Software Engineering (ASE 2005), Long Beach, CA, USA, Nov 2005.

[3] Jeom-Goo Kim Injection Attack Detection using the Removal of SQL Query Attribute Values

[4] R.Ezumalai, G.Aghila Combinatorial Approach for preventing SQL Injection Attacks.

[5] NTAGWABIRA Lambert, KANG Song Lin Use of Query Tokenisation to detect and prevent SQL injection Attacks

[6] Tajpour, A., Massrum, M., Heydari, M.Z. Comparison of SQL injection detection and prevention Techniques

[7] An Intelligent Categorization Engine for Bilingual Web Content Filtering, International Journal of Modeling and Simulation, 7(6), 2005, 1183-1190.

[8]WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation, 34(1), 2008, 65-81.

[9]Log Visualization of Intrusion and Prevention Reverse Proxy Server Against Web Attacks, 70thInternational Conference on Informatics and Creative Multimedia, 2013,325-329.

[10] Seckin Anil Unlu, Kemal Bicakci NoTabNab: Protection against The Tabnabbing Attack.