RESEARCH ARTICLE OPEN ACCESS

Detecting and preventing SQL Injection Attacks

Amir Nasser [1], Reyad Daher [2]
Department of Computer and Control Engineering
University of Tishreen
Latakia - Syria

ABSTRACT

Web applications have opened wide horizons for economic and financial development, sparking hackers' appetites to attack these systems. Most of the attacks on applications and websites belong to the last layer of the OSI, the application layer. These types of attacks can't be prevented by using packet-based firewalls, which examine the signatures and ports used, so it was important and necessary to devise techniques to detect and prevent these attacks, which SQLI attack is the most a dangerous .

The SQLI attack, according to the statistics of information and networks security centers, is the most widespread attack on the application layer, where the distribution areas of this attack are distributed to different countries of the world, , And targets most of the sectors that rely on databases to accomplish services.

The search provides a system to detect and prevent SQLI attacks, using a mechanism that relies on analysis and parsing expression of queries sent to databases associated with applications, to verify the integrity of the query and that it doesn't contain an attempt to inject additional commands that can be used to obtain sensitive users information.

The test of the system presented in the research gave effective results in the detection and prevention of attack, after the implementation of a set of tests, results were examined to know the ability of the system to detect incorrect queries, and study the time delay resulting from the addition of the system provided.

The proposed system was able to achieve 100% of detection and prevention in the tests that have been implemented. The addition of the proposed system to the system under testing led to a delay time, measured between 4 and 6% of the executing time without the system.

Keywords :— Web Applications ،Confidentiality ،Integrity ،Application Level ،SQLI .

I. INTRODUCTION

The growing growth of the Internet has led to the emergence of many services, which have become an integral part of our daily life. Web applications are used to request bookings, pay bills, e-shopping, etc. With the progress and development of e-commerce and services based on Web applications It is necessary to transfer information in a safe and accurate manner.

Most Web applications contain security vulnerabilities that allow attackers to exploit and execute attacks, resulting loss of confidentiality, integrity, and access to information.

The methods and patterns of attacks vary, which requires a variety of protection methods because of the different tools used in the attack and the different level at which attacks are active, each level has it means and tools of protection that is useful to work within it without others.

II. RESEARCH PROBLEM

With the increasing number of websites and applications hosted on them, the number of attacks targeting it increases. Perhaps the most dangerous of these attacks are attacks at the level of applications, specially injection attacks, which exploit the inputs to carry out the attack.

The use of the specific input method means entire depending on the developer of the application to implement protection, which is very risky, while it is not possible to rely on the database management system DBMS (DataBase Mangement System) as a protection tool, because it does not contain the appropriate mechanism to face these attacks and The reliance on firewalls to stop these attacks is not possible because these attacks use ports 80 and 443 that can not be closed by the firewall. Therefore, it was necessary to devise ways and techniques to limit, detect, and prevent such attacks.

III. THE IMPORTANCE OF RESEARCH AND ITS OBJECTIVES

Web applications are an important part of the new e-economy. These applications use bank information for their users from credit card numbers to social security numbers and bank account numbers. Web applications are part of our daily lives that are hard to dispense . Many of them deal with our personal information which may have specific privacy and sensitivity. These applications have started to be exposed to many types of attacks, so the impact of these attacks must be mitigated so that users can continue to benefit from the services of these applications securely.

This research aims to design a system that improves the protection of web applications against one of the most dangerous types of attacks " SQLInjection " attack by accomplishing the detection of attack, and then prevent possible attacks.

IV. RESEARCH METHODS AND MATERIALS

The study was based on a statistical methodology, which examined the dangerous and spread of SQLI attacks based on studies of the most important information and networks security labs to verify the feasibility of the research. In the next stage, an analytical methodology was implemented. To achieve a full and deep understanding of these attacks. At the end of the research, the methodology was applied by constructing a system of detection and prevention of SQLI attacks based on the previous analytical study and then testing the system and studying the test results taking in perspective two parameters :

1. The correct detection rate for non-standard queries. Three type of attacks were carried out, 100 different attacks were repeated for each type , and verification of whether the proposed system could detect these attacks.

2. The delay time of the addition of the analyst (the proposed system) as a stage between the web application and the database. The longer, shorter and the average execution time was calculated by the presence of the system and its absence, and then comparing the obtained values to study the added delay.

V. APPLICATION LAYER ATTACKS

Attacks on Web applications are attacks directed directly at the available Web applications. These attacks attempt to misuse APIs embedded in Web applications, that is, attacks occur within the scope of the application itself. An attacker can use these attacks to:

1. Read, add, delete or modify data.

2. Introducing a virus program that uses computers and software applications to copy viruses across the network.

3- Enter a spy program to analyze your network and gain information that can eventually be used to disable or destroy the system and network.

4 - disable operating systems abnormally.

5. Disable other security controls to enable future attacks.

This type of attack is very popular, and is expected to remain so because most web applications and web services used are typically vulnerable to direct access. According to many statistics, the application layer attacks, specifically the SQLI attack, are ranked at the top in the number of attacks worldwide and compared to the rest of the attacks.

Figure 1 shows according to a report by the European Union Agency for Network and Information Security (ENISA) released in the first month of 2018 that SQLI attacks accounted for 51% of attacks implemented at the application layer level,.

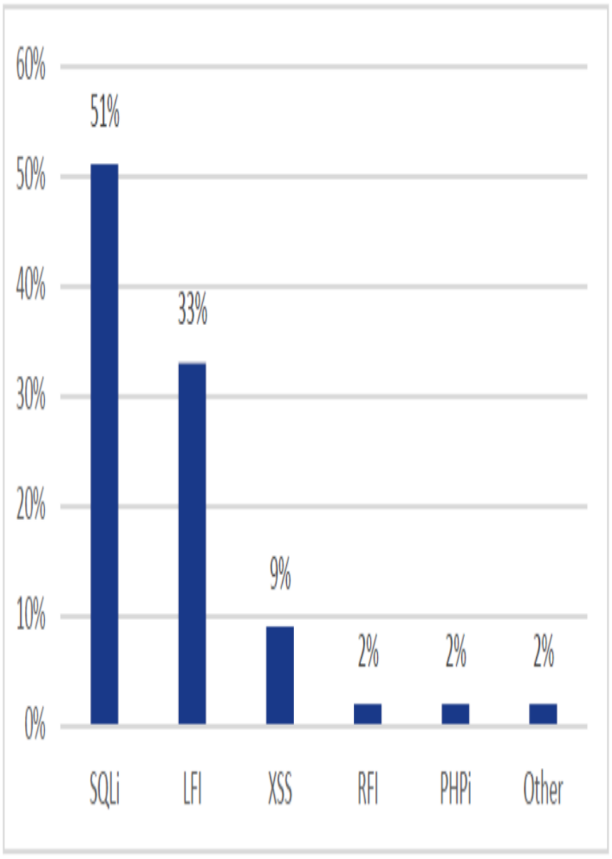

Figure 1 Ratio of SQLI attacks according to ENISA statistics [14].

Application-level attacks vary and can be categorized as follows:

1.code Injection.
2. Broken Authentication and Session Management.
3.Cross-Site Scripting.
4. Insecure Direct Object References.
5. Security Misconfiguration .
6. Sensitive Data Exposure.
7. Missing Function Level Access Control.
8. Cross-Site Request Forgery .
9. Using Components With Known Vulnerabilities .
10. Unvalidated Redirects and Forwards .

VI. SQLI ATTACK

The SQLI attack is intended to exploit applications that use relational databases to get an answer to their requests. The applications create SQL commands and send them to the database. The SQLI attack exploits the fact that many of these applications integrate the fixed value string from SQL commands with user data, which form a where predicates formula or additional subqueries , and in this way SQL

commands are converted to malicious software, which may cause unauthorized access, data deletion or information theft. It is worth noting that most databases can be A target for SQLI attack, SQLI attacks can be categorized in three main categories [2] as in Figure (2) are:

1- order wise
2. blinde
3- against database
These classifications differ in the classification criteria.

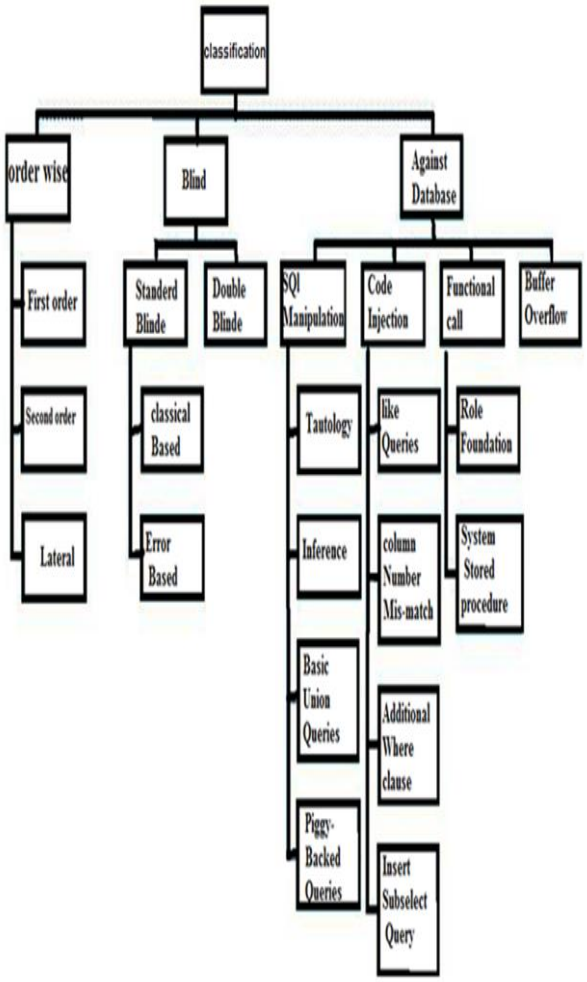

Figure 2 SQLI attack classifications [15]

VII. THE PROPOSED ALGORITHM TO DETECT AND PREVENT SQLI ATTACKS

A web system consists of a user's web browser, where the user is connected to the web application through the Web as shown in Figure 3.

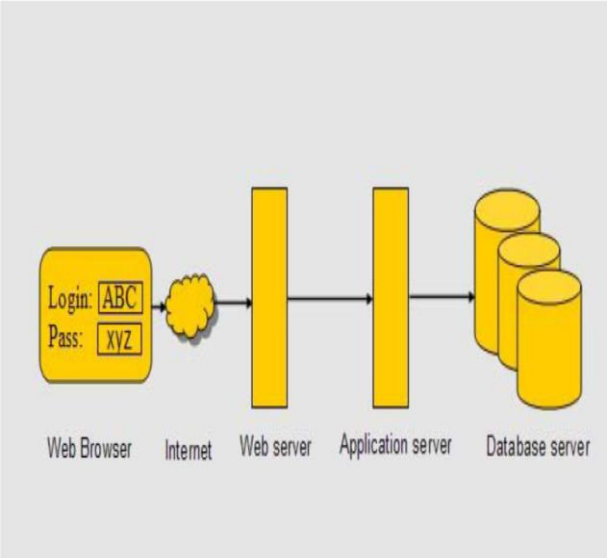

Figure 3 A diagram of the phases of the web system[1]

The vulnerabilities allow attackers to target these applications and obtain high-value or sensitive information. The attacker sends SQL requests to interact with the RDBMS server or modify existing SQL requests to restore unauthorized and unreachable information.
Thus we can see that solving the problem lies in the analysis of the packet sent to the database in search of the injections .So in the research a phase was added to the diagram in Figure (3) to achieve the required algorithm and the web system diagram became a as in Figure 4.

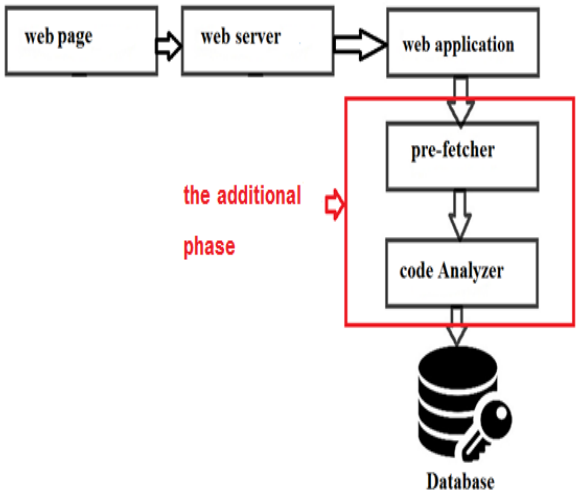

Figure (4): The layout of the phases of the web system after the proposed algorithm

In view of Figure 5, a diagram of the components of this system can be developed to determine its component parts, its sequence of work and its association with each other .

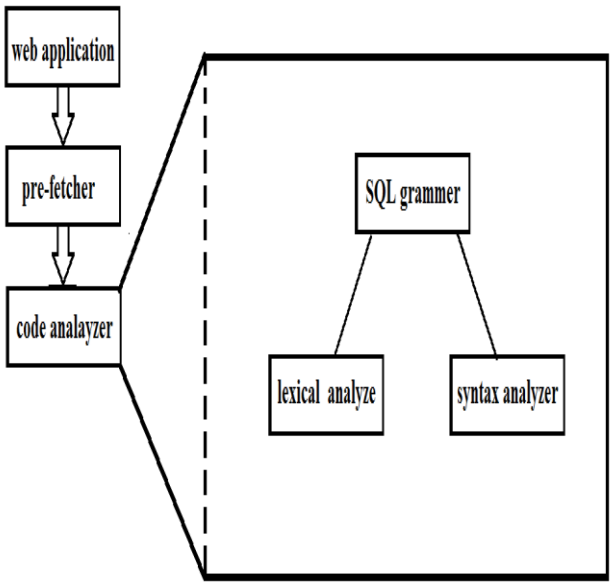

Figure 5: Detailed components of the proposed software system

Thus, it can be seen that the system consists of:

1. Pre-fetcher for queries, which is responsible for capturing queries, and calling the code analyzer instead of sending them directly to the database.

The task of the Pre-fetcher is to prevent the application from sending the generated query automatically based on user input to the database directly, but send it to the code analyzer.

As noted above, the application developer has a role in interacting with the system of software analysis, so it can be very easily Create the Pre-fetcher by adding simple instructions to the application or web page that performs the analyzer's calling as shown in the diagram in Fig 6.

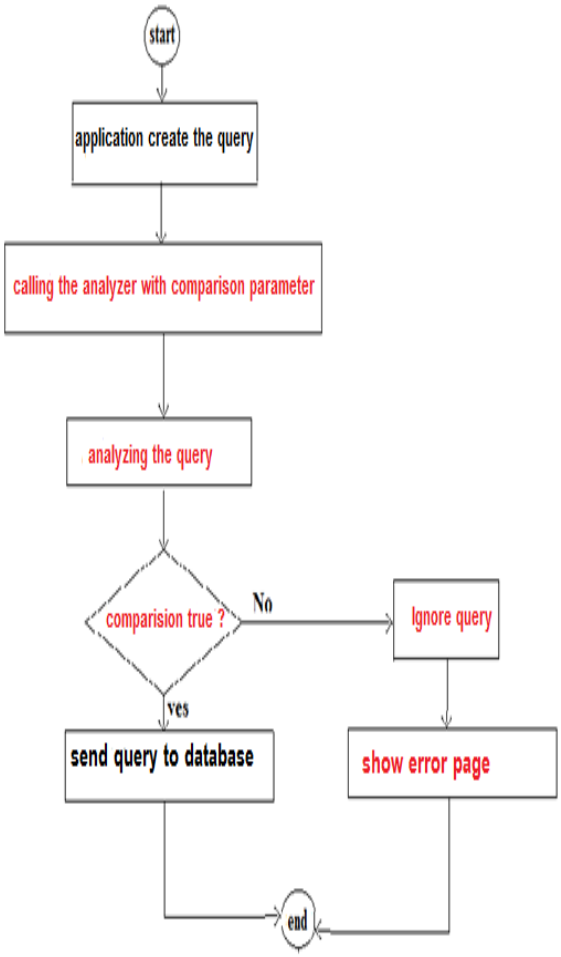

Figure 6: Introducing the analyst call to the application or web page

2- A code analyzer consists of:

1. Syntax Analyzer For the structure analyzer, we have defined the possible permutations in SQL statements , to determine the Syntax of each statement by setting the appropriate grammars with the aim of finally reaching the terminal symbols. As such, when the statement is received, these rules are applied to determine the type of statement And then determine its composition, including the identification of tokens in it

2. lexical analyzer this is where terminal symbols or tokens is the defined , which can be considered as keywords for any statement in SQL language, that is, all the keywords that can be defined in the SQL statements are identified, in addition to specifying the symbols of spaces and moving to the line New or progress in the line “white spaces” so that can be is ignored when parsing the statement .

VIII. MECHANISM OF THE PROPOSED SYSTEM FOR DETECTION AND PREVENTION OF SQLI ATTACKS

The mechanism is how the web application developer interacts with our system, as well as the comparison and inspection mode.

For the pattern of examination, there are two ways:

1- Static:

Static method means that the detection process depends on the construction of a static model of statements , stored in the code analyzer, and then the statement generated from the application is automatically compared with the statically stored statement in the code analyzer, and when the mismatch occurs, the declaration is denied.

2. Dynamic

The language grammars are applied to the query generated in a manner similar to the parsing of the statement. Hence, depending on the machine learning or artificial intelligence, the attack is determined or not, rather than storing a static statement and compare it with the statement generated as in the static method.

In this research, we adopted an intermediate method between the two, where we built a system that analyzes the code based on the rules we presented to it. This is the dynamic part of the work mechanism. It compares it with a parameter that expresses the number of terminal symbols in the correct query to verify Query integrity which is the static part of the mechanism.

As for the interaction of the developer with the system, we have chosen to give the developer a role in the call of the system for the following reasons:

1 - Giving the ability for the developer to call the program allows to make the possibility of integrating the system with pre - built applications easier, so it is just enough to add a call instruction to the system.

2. One of the main reasons for SQLInjection attacks is the weakness developers have in the security aspects when writing their code. Therefore, to understand this point, they must be encouraged to observe the security aspect of their codes by making them a partner in the protection process, knowing that this requires, according to our proposed algorithm, very little knowledge of the concept of tokens and The basics of compilers .

To illustrate the steps of the internal working algorithm of the proposed system, it is possible to use Figure (7), which represents the flow chart for the work of the procedure in a simplified manner.

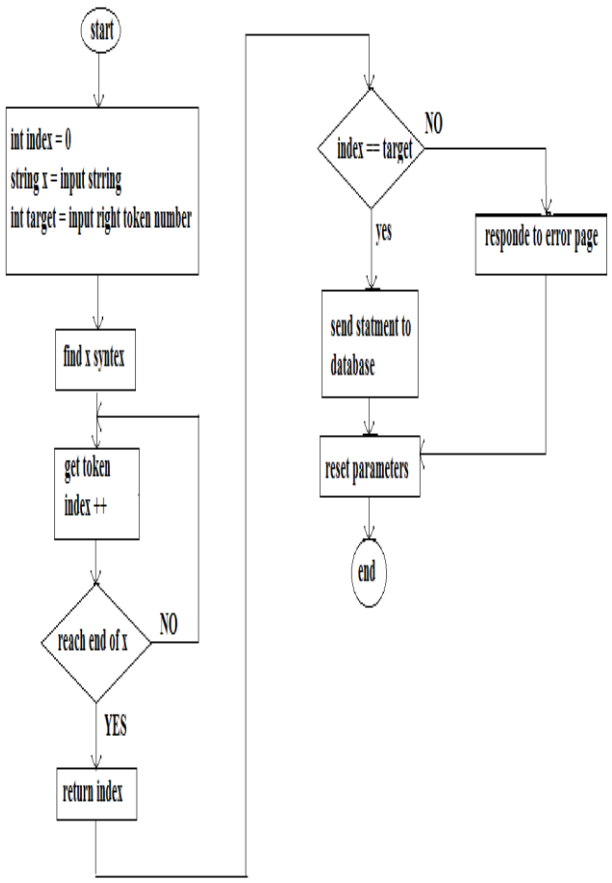

Figure (7): Flow chart of the internal algorithm of the analyst

IX. RESULTS AND DISCUSSION

We have tested a variety of different types of attacks, so we divided the tests into each of them represents a pattern of attacks so we can figure out the detection rate.

Case 1: "WHERE manipulation"

Case 2: "Code injection"

Case 3: "Insert Exploit"

The effectiveness of the proposed system was tested against the three attacks. In each case, 100 SQL queries were tested, each of which contained three different forms of attacks in different and random forms. The analyst was able to detect all these queries.

To estimate the delay that occurred because of the algorithm, we calculated the response time for 100 queries for each of the previous cases separately, and then determine the shortest response time in each case and the longest response time and the average response time for each case, when applying the algorithm, Table 1 shows the results obtained, where time is measured in milliseconds.

Table (1) The delay time resulting from applying the algorithm measured in milliseconds

		shortest response time	longest response time	Response time Average
WHERE manipulation	Without analyzer	725.6	900.1	895.5
	With analyzer	761.9	936.1	940.2
Code injection	Without analyzer	850.4	935.2	900.2
	With analyzer	909.9	1010.6	975.5
Insert Exploit	Without analyzer	750.8	870.4	825.3
	With analyzer	795.8	913.92	885.4

It can be concluded from Table (1) that the percentage of added delay ranges from 4% to 8% and an average of 6%. This is a very acceptable result because the test was performed on one local device and comparing to the real world where response time vary from milliseconds And tens of seconds according to the nature of the application and its work.

X. CONCLUSIONS AND RECOMMENDATIONS

In view of previous results and operating conditions, it is clear that we can say: we have reached our goal of detecting and preventing this type of attack where 100% injections were detected in the tests carried out on the following methods of attacks:

1 - manipulation of where

2 - injected code

3 - exploitation of insert.

Note that the time delay problem caused by the addition of the system as a phase has occurred, as we have resulted in a time increase of between 4-8% of the original execution time without the system.

In terms of compatibility, it is easy to add the code analyzer we built to any current web system working within the Windows environment , since it works as a thirdparty application and therefore there is no need to include its sourcecode or include it as an additional library in the code of the target application protected.

In addition, when considering the complexity of the algorithm provided for the proposed system, and back to Figure (4-5), it is given in relation to 2N + 20, and therefore it is the class O (N) where N is the number of tokens in the SQL query under analysis, It can be classified as a low complication factor compared to complex systems O (N2) or O (N3) in general.

The weakness lies in the application developer who integrates the application. As mentioned earlier, he must have some knowledge of the basic concepts of parsers and compilers. In our view, knowledge of such information has more benefits so The developers may also contribute to mitigating the risk of these attacks.

At the end of the research, the following suggestions can be presented:

1 - rewriting the code using JAVA or Python to ensure compatibility with applications that rely on Linux servers.

2. Develop the working mode to become fully dynamic.

3 - Conduct more tests against other methods of attacks, in order to increase confidence in performance before being put in a real work environment.

ACKNOWLEDGMENT

The research that has led to this work has been supported in part first by the Tishreen University Enterprise .

REFERENCES

[1] SPI Dynamics. Web Application Security Assessment. SPI Dynamics Whitepaper ،2003.

[2] Peter A. Carter . Expert Scripting and Automation for SQL Server DBAs 1st ed .Apress. July 28 ،2016

[3] Mohammad Qbea'h ،Mohammad Alshraideh ،Khair Eddin Sabri. Detecting and Preventing SQL Injection Attacks: A Formal Approach. Cybersecurity and Cyberforensics Conference (CCC) ،2016

[4] Navdeep Kaur ،Parminder Kaur. Modeling a SQL injection attack. Computing for Sustainable Global Development (INDIACom) ،2016 3rd International Conference.2016

[5] Kuisheng Wang ،Yan Hou. Detection method of SQL injection attack in cloud computing environment. Advanced Information Management ،Communicates ، Electronic and Automation Control Conference (IMCEC). 2016

[6] Jai Puneet Singh. Analysis of SQL Injection Detection Techniques. CIISE ،Concordia University ،Montreal ،Qu ´ ebec ،Canada.2017

[7] Arianit Maraj ،Ermir Rogova ، Genc Jakupi ، Xheladin Grajqevci. Testing techniques and analysis of SQL injection attacks. Knowledge Engineering and Applications (ICKEA)، 2nd International Conference. 2017

[8] Maksy Sendiang ،Anritsu Polii ،Jusuf Mappadang. Minimization of SQL injection in scheduling application development. IEEE .2017

[9] Qais Temeiza ،Mohammad Temeiza ،Jamil Itmazi . Enhanced Approach to Detection of SQL Injection Attack . IEEE . 2017

[10] Qais Temeiza ،Mohammad Temeiza ،Jamil Itmazi . A novel method for preventing SQL injection using SHA-1 algorithm and syntax-awareness. IEEE . 2017.

[11] Piyush A. Sonewar ، Sonali D. Thosar . Detection of SQL injection and XSS attacks in three tier web applications . IEEE . 2017

[12] Chen Ping ،Wang Jinshuang ،Pan Lin ،Yu Han . Research and implementation of SQL injection prevention method based on ISR . IEEE . 2017

[13] McAfee Labs Threat Report , December 2017.

[14] ENISA , ENISA Threat Landscape Report 2017 , HSA, January 2018.
[15] http://prog3.com/sbdm/blog/ycdyx/article/details/50218669 . 6/2/2018