System of the SQL Injection Prevention

Voitovych O.P., Kupershtein L.M., Ostapenko A.V., Yuvkovetskyi O.S.
Information Security Department
Vinnytsia National Technical University
Vinnytsia, Ukraine
voytovych.op@gmail.com

***Abstract*— One of the most dangerous vulnerability in cyberspace is SQL Injection. There is known approaches to protect Web-applications against SQL Injection attacks in the article. To improve the web software security it is developed defense mechanism that protects web-resources from SQL-injection performing. The system based on machine learning for preventing SQLi attack is designed. As a result it is received a software tool which allows to protect web software from SQL-injection vulnerability. Developed software tool allows user to protect his own web-application from an attack with using SQL.**

Keywords—Web-resources vulnerabilities; Web-applications; SQL Injection; SQl-atack; prevention syste; machine learning.

I. INTRODUCTION

SQL Injection is one of the most effective methods for stealing the data from the database, with the help of these attacks hackers can get access to the server and steal sensitive information. According to the "Open Web Application Security Project", injection attack is a technique used in hacking or cracking to access information or unauthorized activity [1]. As we have three main methods, described in previous papers, it is necessary to develop more secure method. SQL injection prevention isn't fully secure the database, because every year attacks became more complicated. So none of the proposed method provides the exact solution for preventing injected query [2, 3, 4], there are many additional methods through which injection is possible. In this paper, we design a system based on machine learning for preventing SQLI attack. This system captures HTTP requests to obtain input contents and classifies them by Bayesian classifier, and then detects malicious contents and terminates attacks. In addition, we created a tool for generating training samples automatically through classifying and analyzing legitimate and injection patterns in the real world. We evaluated this learning-based method by various different types of injection patterns, and verified the actual effect with a SQL injection attack tool. Unlike previous approaches, our method is effective with a simple detection mechanism and independent of databases and web applications, in other hands, any web application with any database can be protected by the method well and need not be modified anything.

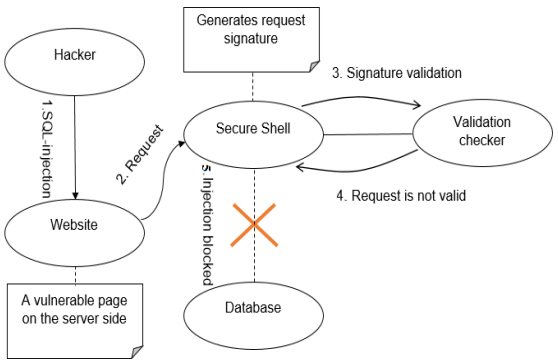
II.SQL INJECTION ATTACK TYPES

Different statistics data shows that more than 40,000 attacks per day happen in real world, so it's a huge problem which needs solution for variety of Web systems [11]. Today there are many Web systems use databases for storing the data needed for the Websites applications activity, as well as user preferences, personal data, sensitive financial information etc. Using variety of technologies allows developers to make Web products interesting and useful for clients (e.g. e-shop, internet banking) [12-15].

SQL injections can be classified into five major categories: Union SQL injections, Error-based SQL injections, Boolean SQL injections, Time-based SQL injections and Out-of-band SQL injections [16, 17].

III.PREVENTION SYSTEM

To protect system against SQL-injection attack we have offered protection (Fig. 1) [11] that performs three functions. When the hacker is trying to use SQL-injection on the vulnerable web-page (1), he forms special request to the server (2). It is necessary to check the information from users, because input data could be dangerous. That's why prevention system has Validation checker. It is a bunch of methods to filter all input data from users (3) that generates request signature. After that injection cannot be used, because Secure Shell filters the output information (4) and blocks injection to the Database (5). Also, prevention system has its own error handler.



Prevention system scheme

The first method is converting special characters to HTML entities. Another two methods are using regular expressions and exceptions.

A. Output escaping.

Because server side of web applications often interacts with web-pages, URL-s and databases, there are a lot of functions that are using to handle this data. Most of the information is handling as a string type, but in the different situations (database or server) it is needed to use different checking with filters. For instance, a space in a web address is specified as %20, while a Single quote (') is specified as *&#039*; There are a number of built-in conversion functions in PHP. That is why it often uses for Output escaping method.

Also it is necessary to check the type or compulsory data type assignment as well as all entered data by performing inspection type, length, format and range data. When implementing precautions against malicious input, it is essential to consider architecture and script execution. For example, if a variable, which has been inserted into the request to be kept numeric data, it is required to use the forced reduction derived type to a numeric. The code is as follows:

B. Check the structure of input data by using regular expressions.

While setting data into a Website users don't always actions as developer provided. Often, users make mistakes. But there are also attackers who try to abuse popular Web systems. For both groups of users, it is important to verify that data what they send is correct. For this purpose could be used a regular expression which is accessible way of determination a pattern of possible strings [18-19].

C. Using exceptions.

To avoid stopping services which happens when Web application specified error occur it is necessary to change the normal flow of the code execution. Such situation is called exception handling. Sometimes software will not execute as it provided to do, resulting in an error. There are a number of reasons that may cause exception, for example: the Web server run out of disk space; a user inputs an incorrect value in a form field; the file or database record is not exist; the software has no permission to do something; the service is temporarily/permanently unavailable.

There are a lot of situations where web-site error could be very dangerous because of information it is shows. In situations like that, developers need to handle the errors. That's why exceptions handling should be part of any programming system. In addition, all the exception should be logged. Based on the severity of an error, notifications should be sent out to other systems/teams.

When an exception is triggered should be provided saving of the current code state. Then the code execution should be delegated to predefined function for exception handling. And this function depending on situation terminates, continues the code running with other condition or resumes the execution from the saved code state.

In this case when the error occurs, our script will stop its run. Also, user will receive a friendly custom error message that doesn't contain any information about source file, stack trace or code.

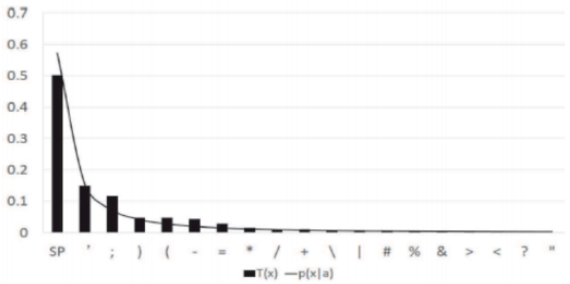
Exceptions are part of nearly every programming language, so it's necessary to use them correctly. In our case we have used them where web-page interacts with database.

D. Prevention based on machine learning

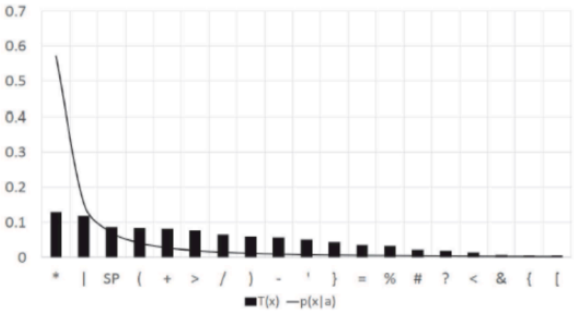
The system based on machine learning for preventing SQLi attack is designed. This system captures HTTP requests to obtain input contents and then detects malicious contents and terminates attacks. We evaluated this learning-based method by various different types of injection patterns, and verified the actual effect with a SQL injection attack tool. Unlike previous approaches, our method is effective with a simple detection mechanism and independent of databases and web applications, in other hands, any web application with any database can be protected by the method well and need not be modified anything.

At first, we have to investigate the keywords used in SQLi attacks. After that, it's important to show the distribution of symbols / keywords.

The curve of Fig. 2 is the distribution approximating the distribution of SQL injection attacks strings. The horizontal axis indicates symbols in the set of attack strings, and the vertical axis indicates the content rate of the corresponding symbols. On the other hand, Fig. 3 shows the distribution of normal strings. We can see that the zeta distribution is not fit well with the distribution of normal strings.



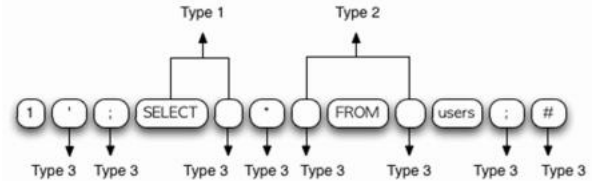
The distribution of SQL injection attack strings



The distribution of SQL normal strings

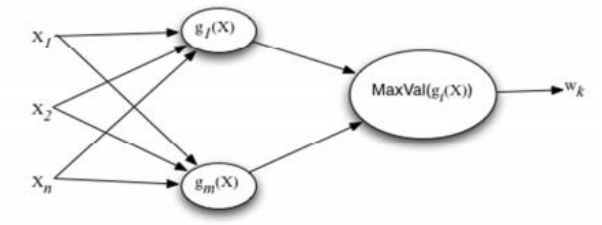
Note that the horizontal axis of the distribution of normal strings is different from the one of the distribution of SQL injection attacks. This means that the appearance symbols between the attack and the normal are differences. In this study, we considered that the feature extraction method of the attack and normal by using the structure equation model which has a latent variables.

It is necessary to convert HTTP parameters into numeric attributes. These attributes are supplied as pattern attributes. We choose the length of parameters and the number of keywords of parameters as pattern attributes. The keywords contain words and symbols in SQL statements, like commas, equal signs, quotation marks, “SELECT”, “UNION” and so on. There are three types of keywords according to SQL statements. The first type must be a blank space on the right side of the word when it is used in any SQL statement, like “SELECT ”; the second one, each side of the word must have a blank space, like “UNION”, and the last, such as commas, need not any blank space aside. Figure 5 shows an example of Converter, the length of this injection code is 24 and the number of keywords is 10.



An example of Converter

In this case it is possible to calculate the probability of each class of an object, and then choose the one that has larger probability as the object' class. The principle of classifier is given in Figure 6.



The principle classifier

In this figure, X is a collection of patterns. If there are n patterns, X is showed as X={(X1, t1), (X2, t2), (X3, t3)...(Xn, tn)}. Xi is the collection of each pattern's attributes, and showed as Xi = (x1, x2...). Ti is the corresponding class name of Xi. Assume that there are m classes, and w means class names, so ti {w1,w2,...,wm}. In practice, Xi has two attributes and m has two classes.

IV. Conclusions

The proposed protection system is designed as a set of scripts that connects to the Web-page resources. This helps to prevent the attack executions via SQL-injection. The approach relies on numeric attributes and training samples. This system has a simple mechanism and can be used for different web applications with different databases. Through some experiments and attacking by a SQL injection attack tool, the results of study show that the system was able to detect the majority of SQL injection techniques and was effective for preventing SQL injection attack.

References

[1] Vulnerability statistics web applications (09.06.2016) http://ptsecurity.ru/download/analitika_web.pdf (in Russian).

[2] Top 10 Vulnerabilities List (19.06.2016). https://www.owasp.org/index. php/Top_10_2013-Top_10 - (in Russian).

[3] A.S. Markov, A.A. Fadin - Systematics vulnerabilities and security defects in software resources (10.06.2016): http://www.npo-echelon.ru/doc/is_taxonomy.pdf. (in Russian).

[4] Common Web Application Vulnerabilities (19.06.2016) https://cve.mitre.org.

[5] "Protecting against exploits in Kaspersky Anti-Virus (15.06.2016): http://www.kaspersky.ru/downloads/pdf/technology_auto_protection_from_exploit.pdf. (in Russian).

[6] O. P. Voitovych, O. S. Yuvkovetskyi. Classification of Web-resources vulnerabilities \\ The 5th International Scientific Conference "Information Technology and Computer Engineering", 2015, pp. 163-164. (in Ukranian).

[7] SQL Injection – OWASP (19.06.2016) https://www.owasp.org/index. php/SQL_Injection. (in Russian).

[8] Top ten most critical Web Application Security Risks, OWASP-Open Web Application Security Project (19.06.2014) https://www.owasp.org/ index.php/Category:OWASP_Top_Ten_Project (in Russian).

[9] Halfond, G. William, J. Viegas, A. Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE International Symposium on Secure Software Engineering. Vol. 1. IEEE, 2006, pp. 13-20.

[10] L-K. Shar, H-B Kuan. "Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns." Information and Software Technology, 2013, pp.1767-1780.

[11] Voitovych, O. P.; Yuvkovetskyi, O. S.; Kupershtein, L. M. SQL Injection prevention system. In: Radio Electronics & Info Communications (UkrMiCo), 2016 International Conference. IEEE, 2016. p. 1-4.

[12] Li Qian, Zhenyuan Zhu, lun Hu, Shuying Liu “Research of SQL Injection Attack and PreventionTechnology 2015” International Conference on Estimation, Detection and Information Fusion (ICEDIF 2015), pp. 303-306.

[13] Sadeghian, Mazdak Zamani, Azizah Abd Manaf. "A taxonomy of SQL injection detection and prevention techniques. International Conference on. IEEE Informatics and Creative Multimedia pp. 234 – 245.

[14] Understanding SQL Injection (26.06.2016) http://www.cisco.com/c/ en/us/about/security-center/sql-injection.html.

[15] T. Atefeh, M. Massrum, M. Zaman. "Comparison of SQL injection detection and prevention techniques" 2nd International Conference on Education Technology and Computer, Vol. 5, 2010, pp.348-359.

[16] A design review: Concepts for mitigating SQL injection attacks." 4th International Symposium on Digital Forensic and Security (ISDFS)». 2016, 169 p.

[17] J. Clarke. “SQL Injection Attacks and Defense. Second Edition”, Syngress, 2012, 576 p.

[18] Wan Min, Kun Liu. “An Improved Eliminating SQL Injection Attacks Based Regular Expressions Matching.” International Conference on Control Engineering and Communication Technology (ICCECT), 2012, 278-289 pp.

[19] X. Qian., H. Peng. “On Defense and Detection of SQL SERVER Injection Attack”, 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2011, pp. 1-4.