CLASSIFICATION ALGORITHM: WEB FIREWALL

Muhammad Saidu Aliero [1], Bilyaminu Isah Shamaki [2], Abdulazeez Muhammad Bello [3], Ibrahim abubakar[4], Bashar Umar Kangiwa[5]

[1] ICT, Kebbi State University of Science and Technology Aliero, Nigeria
[2]National Space Research and Development Agency, Nigeria
[3]Waziri umaru federal polytechnic birnin kebbi, Nigeria
[4]Asset management department Nigerian deposit insurance corporation, Nigeria
[5]Department of computer science, Kebbi State University of Science and Technology Aliero, Nigeria

Abstract

SQL Injection Attack (SQLIA) is one of the most severe attacks that can be used against web database driven applications. Attackers' use SQLIA to get unauthorized access and perform un-authorize data modification as result of improper input validation by web application developers. Various studies have shown that average of 64% of web application of worldwide are vulnerable to SQLIA as result of their vulnerability.

To mitigate the devastating problem of SQLIA, this research proposed web application firewall for SQL injection Attack (SQLIA) that protects unauthorized users from SQLIA. Recent study shows that there is need for improving effectiveness of existing SQLIA firewall to reduce the Loss of data, getting vital information and risk of being attack as result of inaccurate false negative and false positive result reported by the SQLIA. The research focus on improving effectiveness of SQLIA firewall by proposing web application firewall for blind and tautology SQLIA in order to help minimizing of false positive an false negative result as well as to provide the room for improving proposed SQLIA by the potential researchers.

To test and validate the accuracy of research work, three vulnerable web applications were developed with different type of vulnerabilities and accuracy metric were used to analyze the result of three experiments. The result of analysis shows significant improvement by achieving 88.8%. Accuracy for the first experiment, 77% accuracy for the second experiment and 73% accuracy for the third experiment and overall of 79.6%.

Keyword: Classification, web firewall, web security, injection attacks SQL injection

1.INTRODUCTION

Web applicatios are associated with diffirent types of vulnerabilities such as Cross- Site –Scritping (XSS), SQL Injection, File Includion, Brute Force among other vulnerabilities. The most common techniques by web application can be prevented against malacious request is to deployed web application firewalls. A web firewall is a system for detecting of web application attacks. Web firewalls are used for a variety of purposes. Most prominently, they are one of the main barriers between stored database and client accessing the data to prevent it from SQL injection attacks. SQL injection attack is attacks can be performed against web driving database application to execute un-authorized data manipulations and retrievals.

Web firewall can be use as barrier against SQL injection attacks. Most of studies argue that the best approaches by which filter can be applied to differentiate- between malicious and valid request to application, such as blacklisting, whitelisting, pattern matching. However, attack score is getting increase every year regardless of firewall deployed in various applications. As suggested by many studies, this could be because of technological advancement and technical logic of the attackers every day new attack patterns are constructed to bypass existing firewalls as well as many of the deployed firewall are not effective enough to detect existing and newly constructed attacks.

A logical approach to tackle this problem is, to deploy web firewall to block malicious request. There are numbers of commercial as well as open web application

firewall available to perform security test, detection and prevention. However, most of these firewall tools have problem regarding low coverage detection and reporting high percentage of false positive attacks.

2.OVERVIEW OF SQLIA WEB FIREWALL DESIGNING APPROACH

Design is the process of transforming all information gathered and structured in phase 1 into concrete idea about the new or replacement of new information system. It's not recommended to start coding a new system without having demented details on how system component are brought together, how different component of system interact, and classifying dependent and non- depended components. This section provide the details architecture, activities and algorithm design of propose web firewall.

2.1. Architecture of Proposed SQLI Web Firewall

Proposed SQLIA web firewall architecture consist of six components event interceptor, tokenization, parser, abstract syntax tree generation, pattern matching and classifier. These components represent basic fundamental elements for structuring proposed SQLIA web firewall. During implementation stage these components are presented in form of modules, classes, objects or as a set of related function. The SQLIA web firewall will consist of component that include, attack pattern, database of malicious SQL keywords library, classifier component. Figure 4.1show architecture of propose SQLIA web firewall.

I. When user provide seed URL, the first component called event interceptor will filter all query sent to the web application for malicious request examination.
II. The second component called tokenizer which breaks user query into chunk of SQL keywords the purpose to generate syntax three that will ease the attack pattern matching activities.
III. The third component is parser which parses blocks of keywords to syntax tree generator which generate SQL query tree like structure to identify strange or unwanted keywords in user query.
IV. The fourth component abstract syntax tree which represent Source Code as a tree of nodes representing constants or variables (leaves) and operators or statements (inner nodes). Also called a "parse tree". An Abstract SyntaxTree is often the output of a parser (or the "parse stage" of a compiler), and forms the input to semantic analysis and code generation (this assumes a phased compiler; many compilers interleave the phases in order to conserve memory).

Unlike concrete syntax, which consists of a linear sequence of characters and/or tokens, along with a set of rules for parsing them, abstract syntax doesn't (generally) have to worry about issues such as parser ambiguity, operator precedence, etc.

The fifth component is pattern matching that checking a given sequence of tokens for the presence of the constituents of some pattern. In contrast to pattern recognition, the match usually has to be exact. The patterns generally have the form of either sequences or tree structures. Uses of pattern matching include outputting the locations (if any) of a pattern within a token sequence, to output some component of the matched pattern, and to substitute the matching pattern with some other token sequence (i.e., search and replace).

Sequence patterns (e.g., a text string) are often described using regular expressions and matched using techniques such as backtracking.

Tree patterns are used in some programming languages as a general tool to process data based on its structure, e.g., Haskell, ML, Scala and the symbolic mathematics language Mathematical have special syntax for expressing tree patterns and a language construct for conditional execution and value retrieval based on it. For simplicity and efficiency reasons, these tree patterns lack some features that are available in regular expressions.

Often it is possible to give alternative patterns that are tried one by one, which yields a powerful conditional programming construct. Pattern matching sometimes includes support for guards.

Term rewriting and graph rewriting languages rely on pattern matching for the fundamental way a program evaluates into a result.

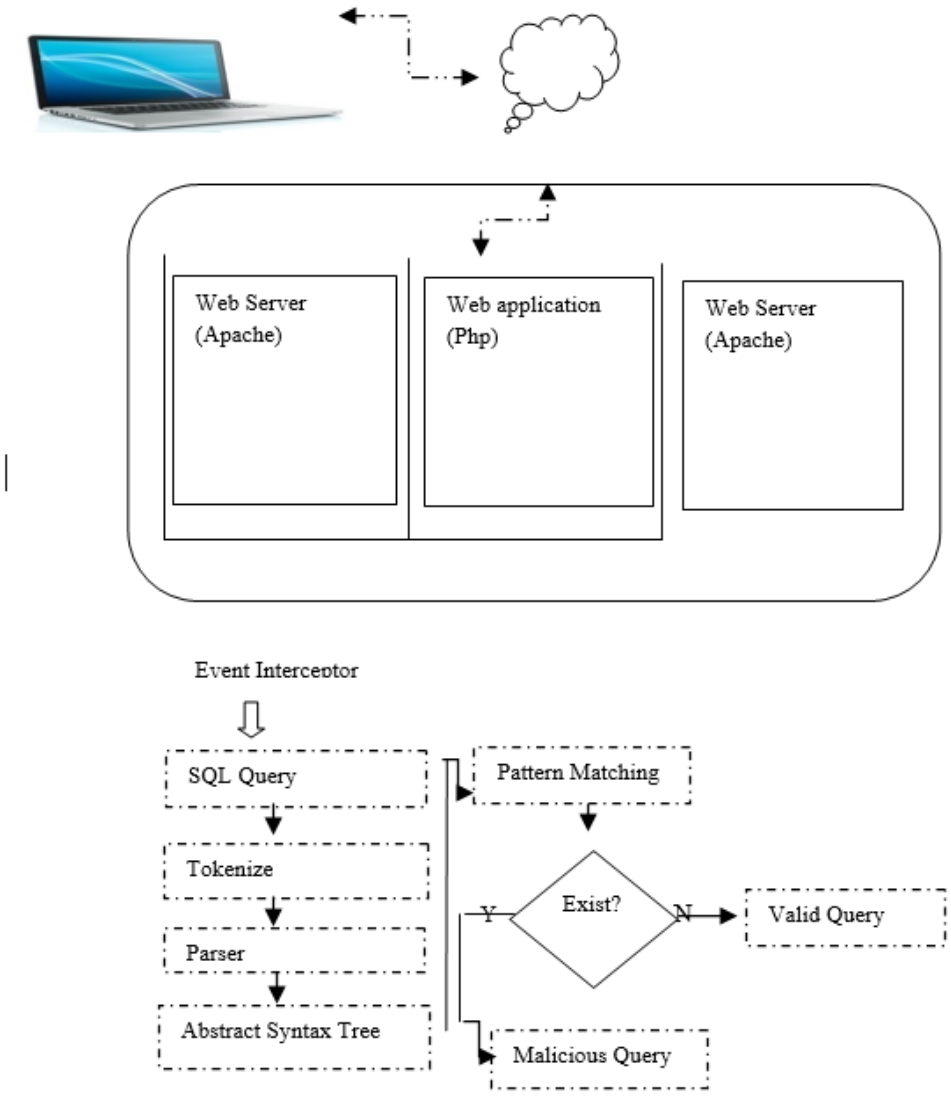


Fig2. Architecture of propose sqlia firewall

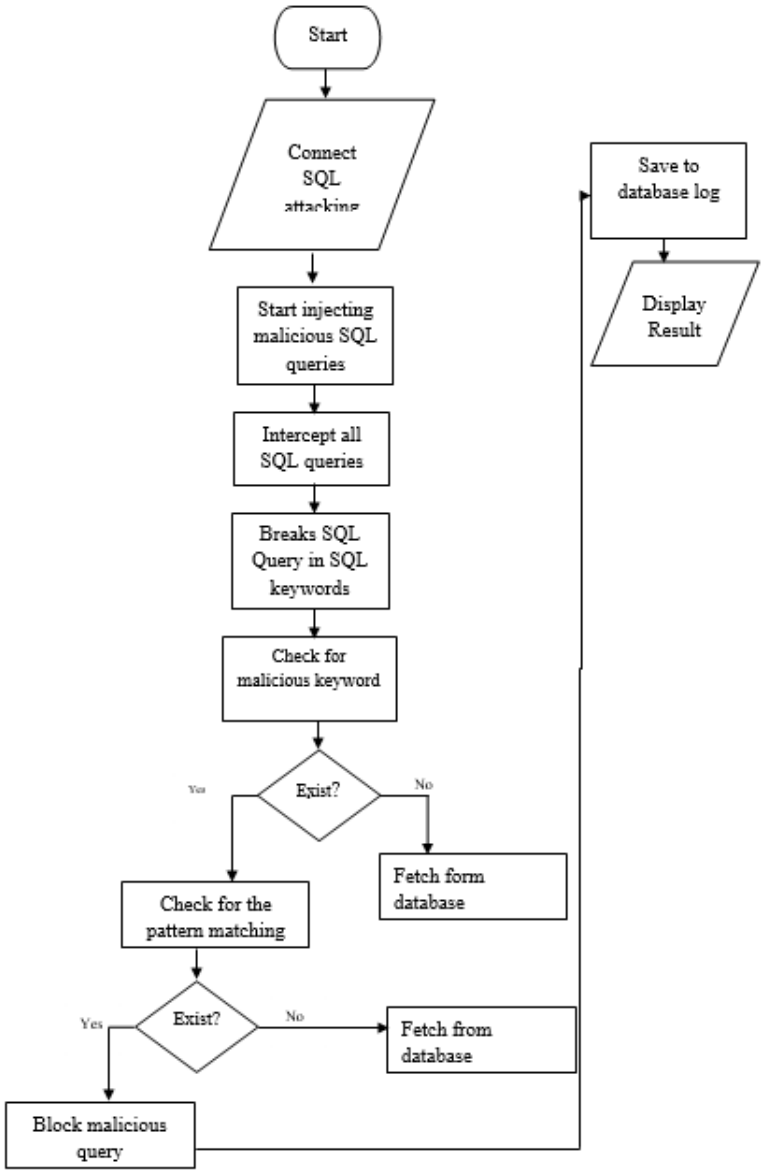


Fig. Algorithm for Proposed SQLIA Web Firewall

2.2. Algorithm of Proposed SQLIA firewall

Algorithm design is one of the fundamental element in software design, it describe steps, procedure, sequence, variable and decision that needed to brought designed software into reality. Having algorithm designed enable developer to examine and image the solution in more concrete manner. To make proposed SQLIF easier to implement its component need to need to be divided into series of phase of processes and decision. This will reduce the complexity of implementing designed algorithm. Figure 4.2 below represent the description of logical procedure proposed SQLI Firewall undergoes to discover SQLIA in a target application.

3. EXPERIMENT AND RESULT DISCUSSION

Proposed SQLIA web application firewall implement pattern matching approach is required to understand how SQL query is constructed is primary for efficient pattern matching of malicious queries, in effective way. Many of the reviewed SQLIA firewalls fails to identify malicious query that has human like's valid names. For example consider web application that requires login

authentication, when users wants to use the system has to provide the login credential if it happens the user name has some attack pattern like in his name the firewall will consider that request to use the system as malicious query which is valid query.

In figure 5.1 shows the scenario where an attack tries to exploit the system to get unauthorize access to the system. As shown in the injection parameters those attacks patterns will automatically blocked by proposed SQLIA web application firewalls which almost most the review web application firewalls are capable of.

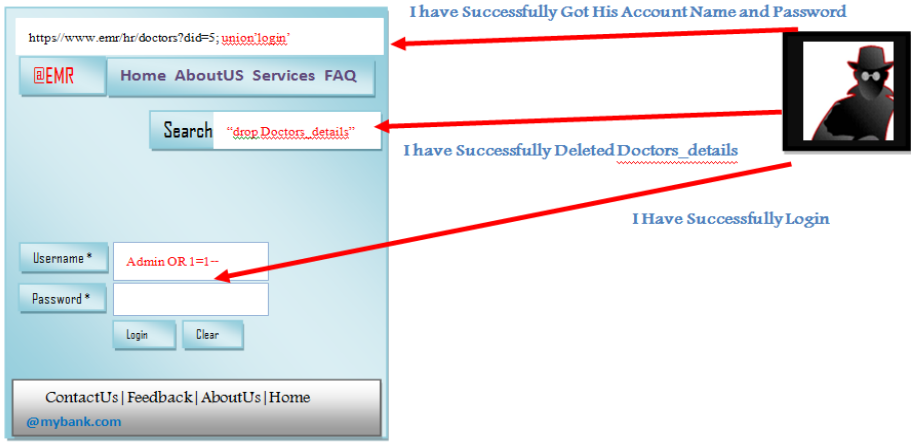


Figure 5. 1 Example of SQLI attack patterns injected in three injection parameters

But the problem arise when user with names like Orton, Anderson and other names that has SQLIA pattern like in their names try to use the system. As shown in figure 5.2 which most of the review web application have problem of differentiating from been a valid query.

Figure 5.2 Valid user request which consider as an attack by many reviewed SQIAWF

As can be seen user Orton try to use the system and his request to use the system will be blocked by many of the reviewed web application firewall which is different from proposed web application firewall proposed in this research work.

Vulnerable Target Applications Used

Despite there are number of vulnerable applications designed to allow individual or vendor to validate their work against attacking tools the research choose to designed three custom Web applications (See Table 5.5). One of the reasons in that most of the related web application firewall studied in literature review are tested in different scenarios and different security configuration type therefore , the research chooses to develop these application to simulate these scenarios so that each proposed work can be evaluated based on its original secure web application. Another reason is that most of the individual or vendors adjust the effectiveness of their tool with respect to security configuration in their proposed work which may not predict effectiveness of the tool in other application as

different developer have different ways of writing same query(Antunes and Vieira, 2010), (Djuric, 2013).
The First target application is online human resource (OHR) application consisting of nine 9 known SQLIA (See Table 5.3) three error based, one blind SQLIA and two 2 vulnerable login query placed our proposed barrier to defend against attacking tool which was developed by our friend . We choose to create more vulnerable pages on this target because OHR contain almost any information regarding employee information in an organization which results in having multiple queries about employee information for different purpose. Similarly this application has two different login queries this is because to differentiate between normal employee accesses to the system with administrator access to the system and we know that each department in organization has one administrator.
The second vulnerable target application is online birds farming application with nine 9 known SQLIA (See Table 5.3) three (3) blind SQLIA and two vulnerable login authentication queries.

The third vulnerable application is online news application with four (4) know SQLIA (See Table 5.3) one(1) error based SQLIA and one (1) vulnerable login authentication and two blind SQL injection firewall. The different between this application and other two vulnerable applications is that in this a query application (vulnerable blind SQL injection attack) was designed to perform information request with of HTTP GET parameter without using any form input tag. This practice is mostly found in news website. for example you may find online news website that provides description or headline of the news but the actual or remaining part of the news is stored in database. When user click on "read more" button then the content of the news headline is loaded and display to the user. This type of query is vulnerable to SQLIA.

All three vulnerable target applications are configure with our SQLIA web application firewall running on window 7 32 bit operating system and 6GB Ram, first and second application running on apache 2.4 with MySQL 5.5.19, and third application running on Apache 2.2.3 with MySQL
5.0.77.

3.1. Experiment 1

The first experiment was carried out on Online Human Resource Application (OHR) which contain similar vulnerability used in testing MySQLinjector and other added vulnerabilities that original author of the scanner do not include. Propose SQLIA web firewall was able to intercept queries, tokenize them, and perform malicious SQL keywords identification and pattern matching activities. As can be seen below (see Figure 5.3) the input URL of OHR application is given to attacking scanner and its display the result (See Figure 5.4).
Propose SQLIA web application firewall identified five out of six SQLIA injected attacking scanner in HR application.
This is because propose web application firewall uses to phase of malicious SQLIA detection and prevention approach that enable propose SQLIA web firewall to conclude weather the query is an attack or valid.
Beside this proposed SQLIA web firewall can prevent against blind SQLIA and block tautology SQLIA of various patterns which present big challenge by previous works. However propose SQLIA web application firewall failed to block attacks of the tautology SQLIA. In this experiment propose SQLIA web application firewall achieved 88% accuracy on two different types of SQLIA (tautology and blind SQLIA) when measure using accuracy metrics (see Section 5.4.1 in Figure 5.5)

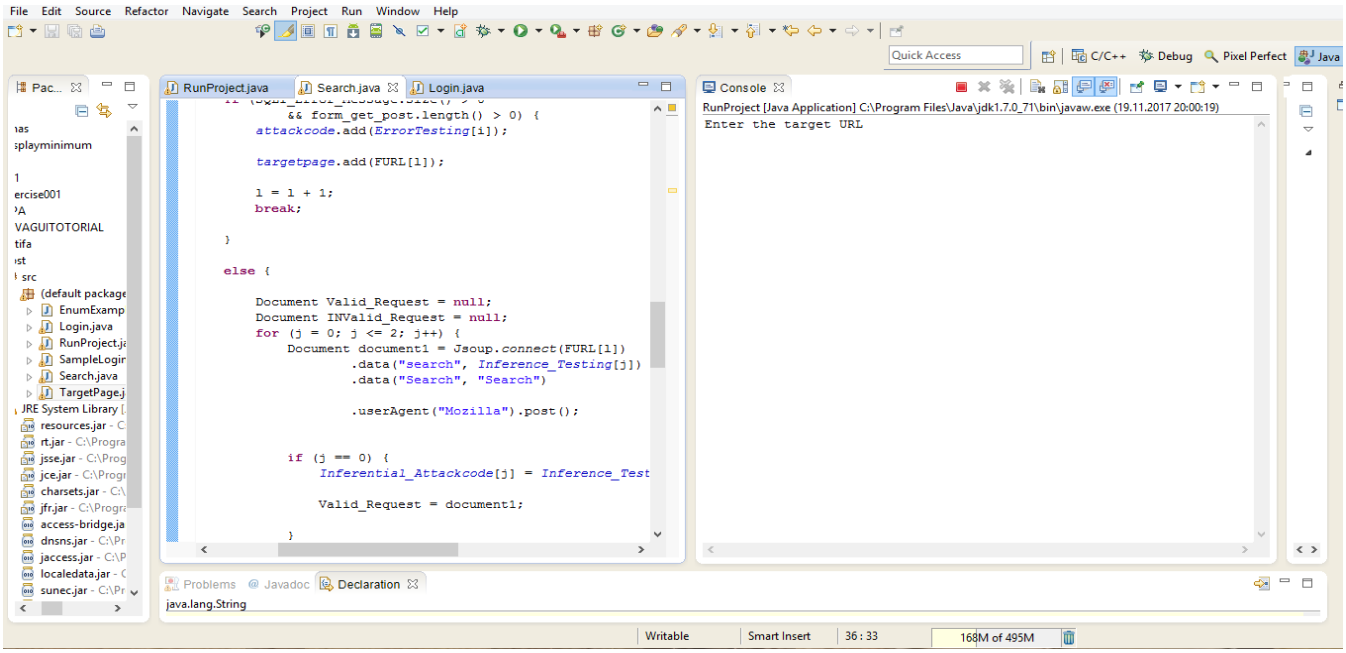


Figure 5.3: Input URL for the OHR vulnerable application

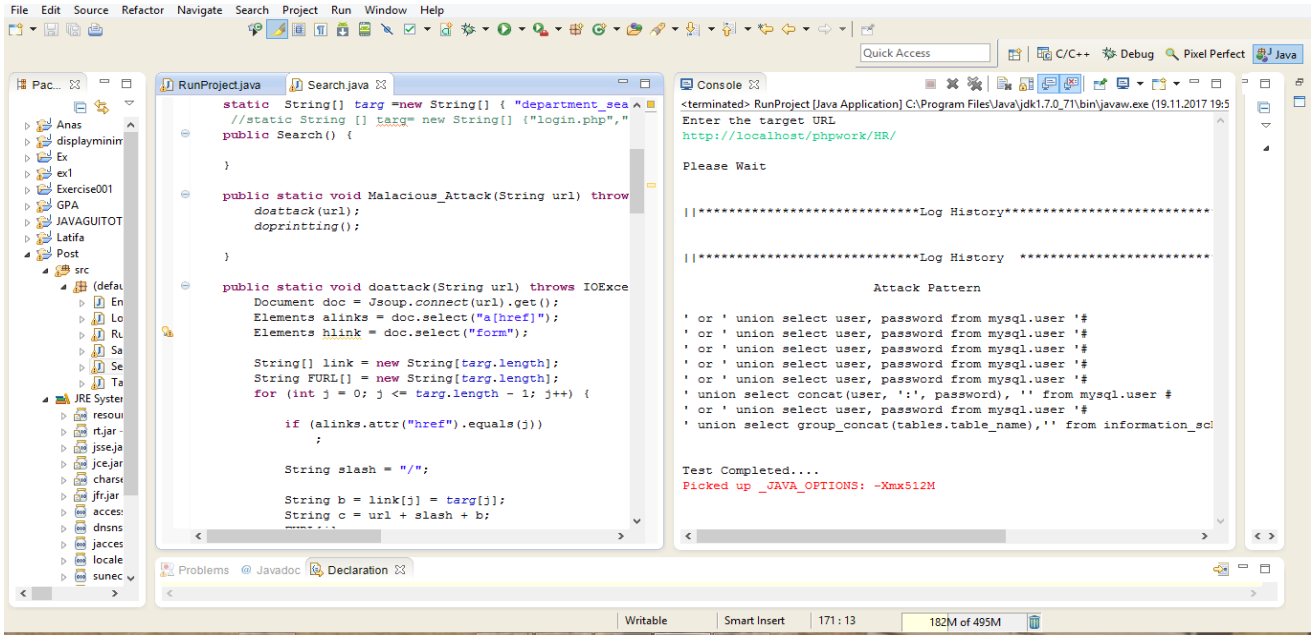


Figure 5.4: defensive result of OHR application

3.1.1. Result Analysis

This section present metrics use to evaluate the accuracy of propose SQLIV scanner. This analysis is similar to the one used in evaluating previous work

ACCURACY FOR EXPERIMENT 1= TPA/TKA * 100, FPA/TKN *100

Total number of true positive attacks (TPA) =8, Total number of known (TKA) =9 Therefore, ACCURACY = 8/9 * (100%), =88%

Total number of false positive Attack (FPV) =1, Total number of known attacks (TKA)=6 Therefore, ACCURACY = 1/9 * (100%) , =11%

Propose SQLIA web application firewall achieved 83.3% coverage of true positive and 11% false positive compare in (Diksha G. Kumar and Madhumita Chatterjee, 2014) which achieved 81.7% coverage of true positive and 18.3% false positive. Similarly proposed SQLIA web application firewall achieved less coverage of true positive attacks when compare with (Baohua Hung, Tongyi Xie and Yan Ma, 2015) achieved 79% and missing only 21% of vulnerabilities on target applications. Because the authors in (Hossain Shahriar and Mohammad Zulkernine, 2015) claimed to update the database of attacks patterns used by thereby producing more accurate result compare to previous work which achieved 82%.

3.2 Experiment 2

Second experiment was on Farm Online application consisting of five (5) vulnerable queries: two login queries (" admin.php "and "Login.php"), three vulnerable search queries This experimental setup is similar to setup used in OHR vulnerable application the only difference is in query used here have the catching approach to return custom error messages when abnormal query is received by the database server. Similarly in HR the login.php query is designed to connect the ser/employee to application if the query return at least one record (which is easy to bypass using tautology SQL injection attack) while in farm application the login.php is more difficult to bypass using simple tautology attack because it compares not number of true record return but actual rows returned by the query. To prevent above mentioned SQLIA it is required for propose SQLIA web application firewall to perform careful tokenization of SQL query in effective way. In this case propose SQLIA web application firewall pattern matching to deduce query with high potential to been malicious in target web application. Propose SQLIA web application firewall successfully blocked tautology SQLIA injected by attacking scanner. As can be seen below (See Figure 5.5) input URL of Farm application is given to attacking scanner and proposed SQLIA web application firewall successfully identified total number of four (4) SQL injection attacks send by attacking scanner out of five as display in Figure 5.6. In this experiment propose SQLIA web application firewall achieved 80% accuracy when measure using accuracy metrics (see section 5.5.1 and in Figure 5.7)

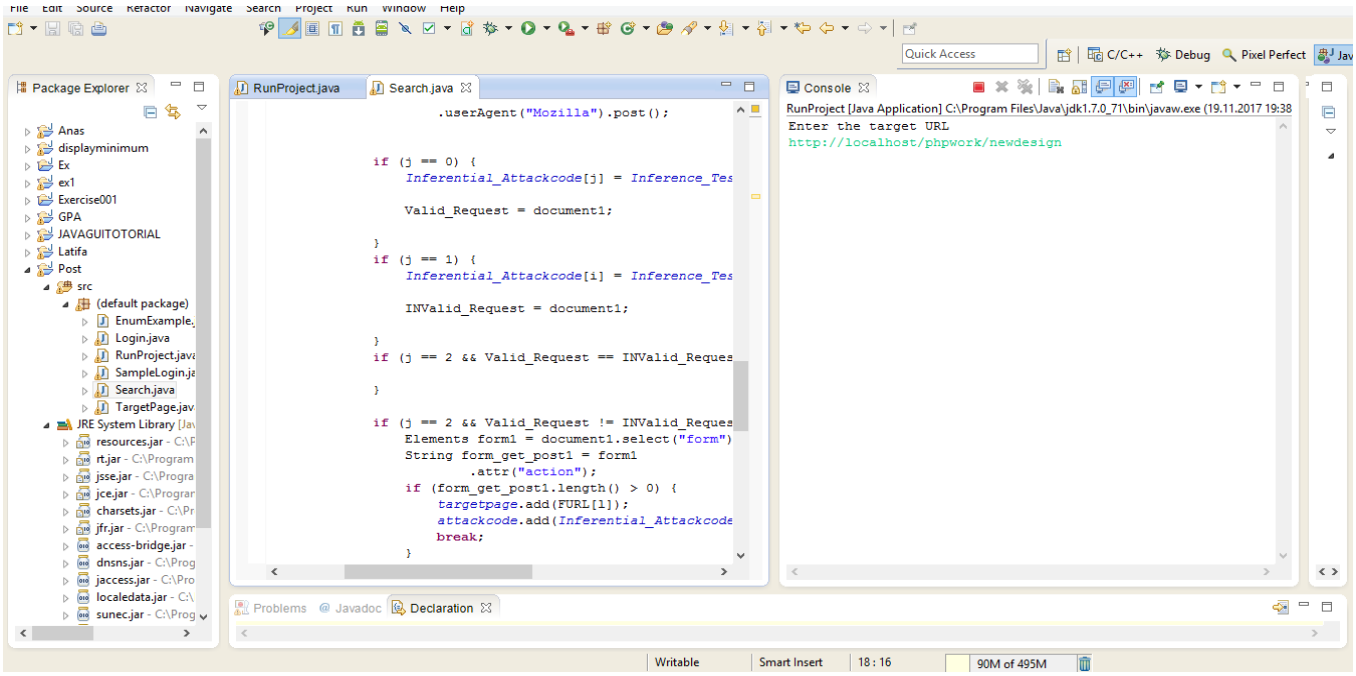


Figure 5.4: Input URL for the Newdesign vulnerable application

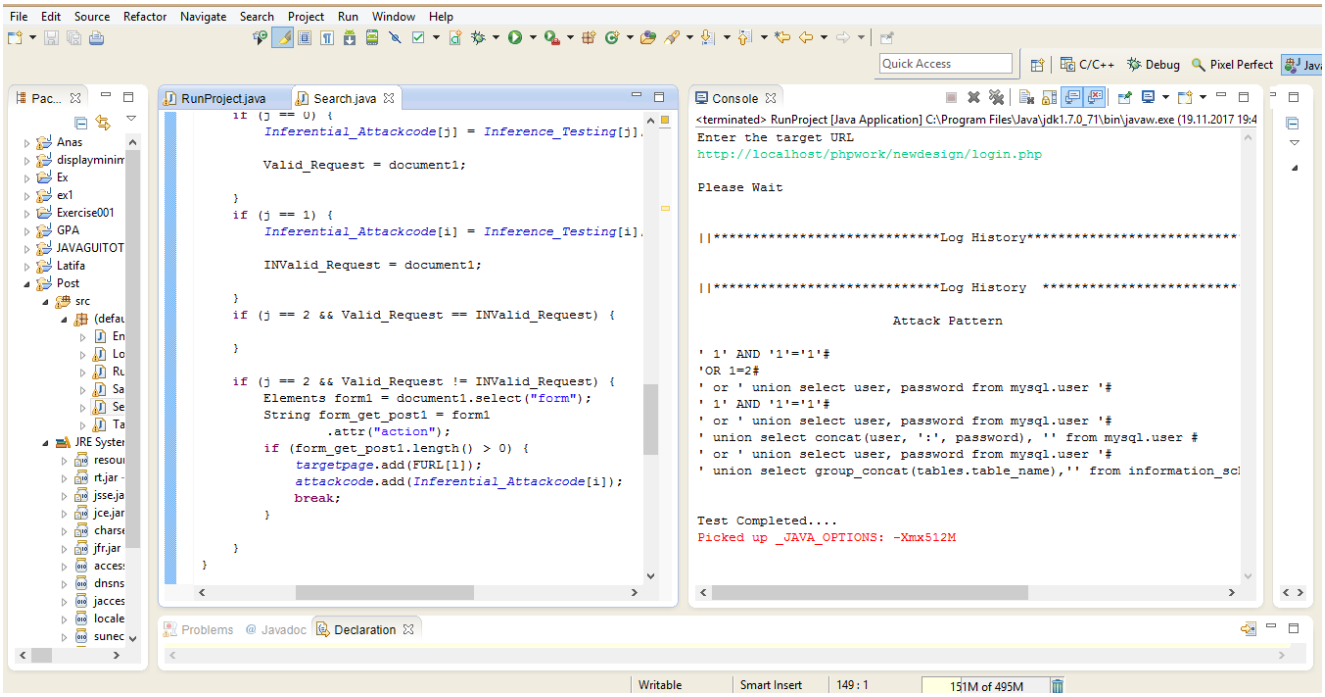


Figure 5.5: defensive result of Newdesign vulnerable application

3.2.1. Result Analysis

This section present analysis result similar to the experimental setup used (Diksha G. Kumar and Madhumita Chatterjee, 2014), (Hossain Shahriar and Mohammad Zulkernine, 2015) previously compare with propose proposed SQLIA web application firewall (see section 5.4.1) however there is little difference between how these two applications (OHR and Farm) are designed in their queries (see section 5.4) . As mention earlier that both previous methods are only effective if target application attacks user's name do not have any similarities with SQLIA pattern defined in database of attacks patterns libraries. It's not surprise to (Hossain Shahriar and Mohammad Zulkernine, 2015) achieved 69.3%. This is because these scanners are tested on application that is configure to block only attacks that has SQLIA attacks pattern which is statistically define in database of SQLIA.

Previous web application firewall missed such kind of attack in a target application; therefore, the focus of research in this experiment is to improve the pattern matching in previous work Diksha G. Kumar and Madhumita Chatterjee, 2015) by achieving 60% to be able to differentiate between valid users query and malicious users queries.. Result of this analysis (see Figure 5.7) shows that proposed SQLIA web application fire wall achieved 77% accuracy and misses 23% of injected attacks by SQLI attacking scanner on tested application.

Total number of true positive attacks (TPA) =4,
Total number of known attacks (TKA) =5
Therefore, ACCURACY = 7/9 * (100%), =77%
Total number of false positive attack (FPA) =2,
Total number of known vulnerability (TKV) =5
Therefore, ACCURACY = 2/9 * (100%), =23%

3.3. Experiment 3

The third experiment is on Online News vulnerable application which is similar to scenario proposed by (Hossain Shahriar and Mohammad Zulkernine, 2015). Propose SQLIA web application firewall successfully block three malicious queries injected by attacking scanner in news application URL is given (see Figure 5.8) but failed to identified one SQLIA as shown in below (see Figure 5.9). This is because propose SQLIA web

application firewall tokenizer does not have intelligence to learn from future attack pattern which is not include in database of attacks pattern libraries is using one of the injection parameters (HTTP GET) propose SQLIA web application firewall claim to be based on (see section 5.2). Including this type of scenario is time consuming and need to be done carefully otherwise high false negative result will be produced. In this experiment propose SQLIA web application firewall achieved 75% (see Figure 5.10) accuracy when compare using accuracy metrics.

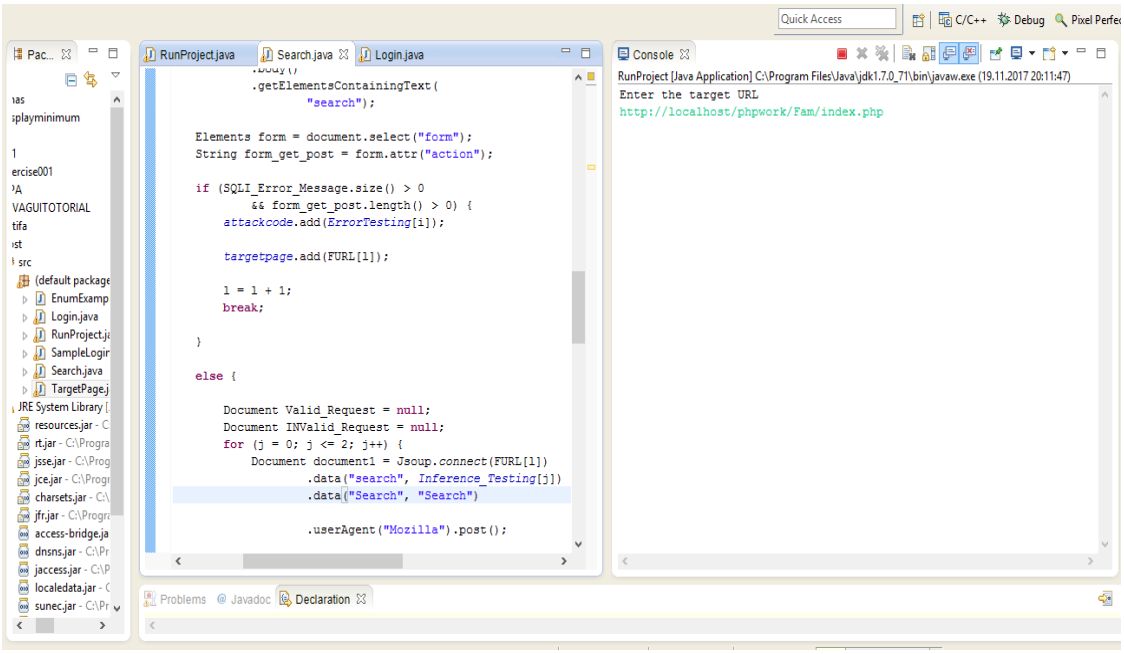


Figure 5.6: Input URL for the Farm vulnerable application

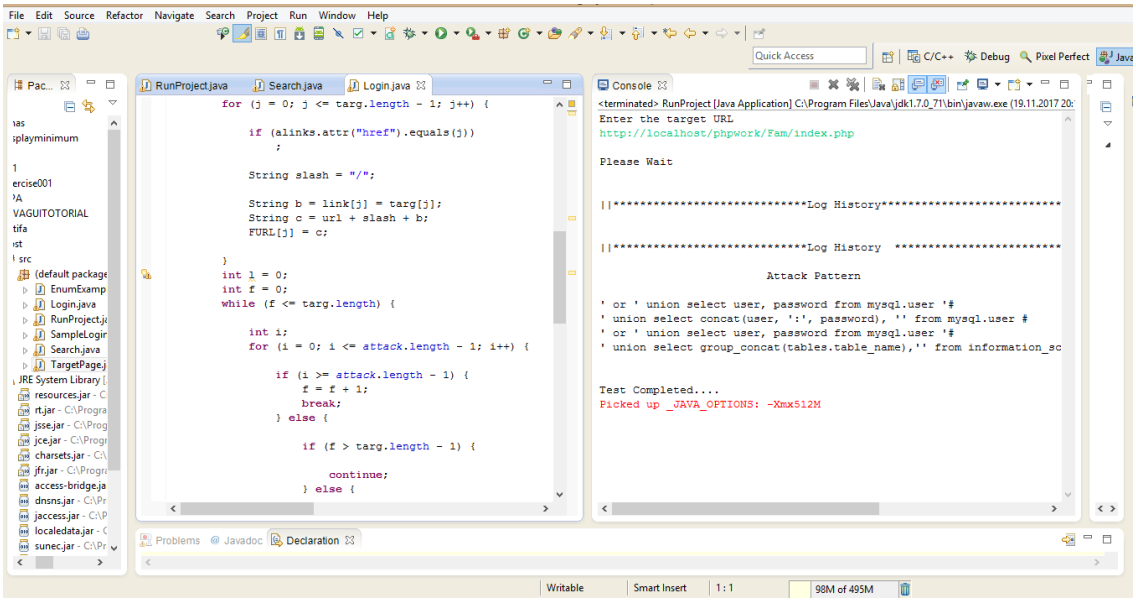


Figure 5.7: defensive result of Farm application

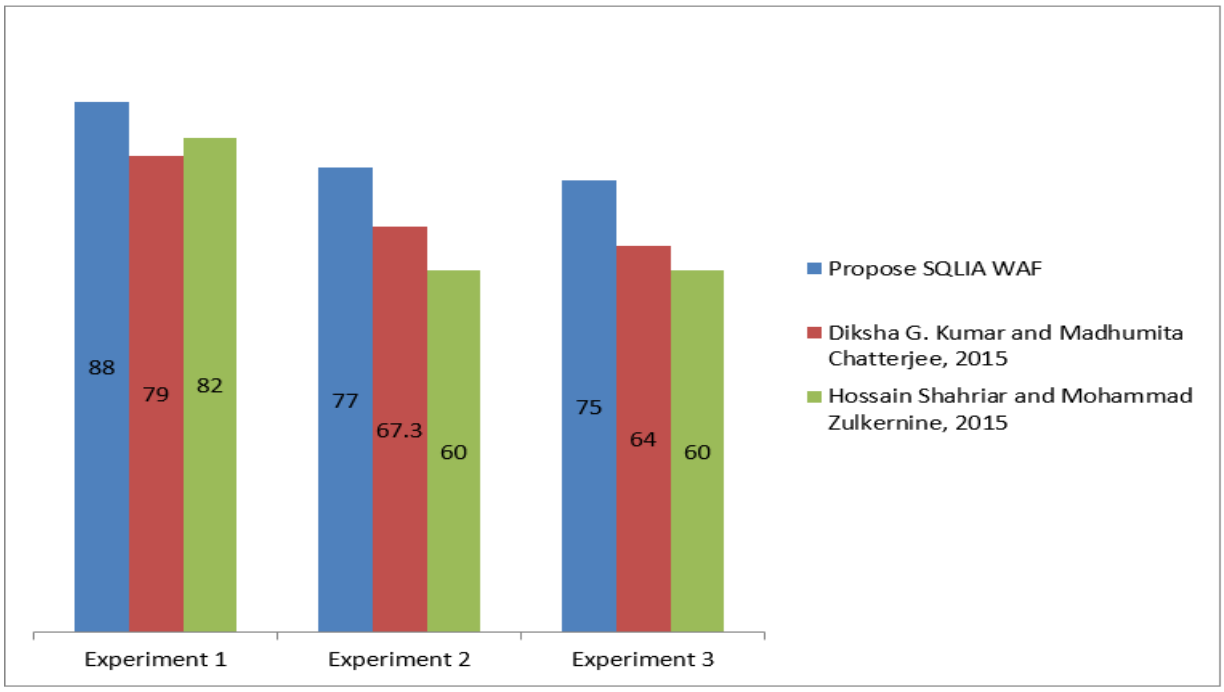
3.3.1 Result Analysis

This section present analysis result of proposed SQLIA web firewall. This setup is different from other two setup (OHR and Farm setup) as explained earlier (see section 5.3).LIVS proposed in (Djuric, 2013) uses three vulnerability analysis component as our propose firewall do. The fore, the focus in this experiment is to improve accuracy in (Hossain Shahriar and Mohammad Zulkernine, 2015).

Experimental result shows that web application firewall in to (Hossain Shahriar and Mohammad Zulkernine, 2015) achieved 64% accuracy while reporting 36% false positive. In this experiment proposed SQLIA web application firewall achieved 75% accuracy and report with 25% false positive (see Figure 5.10). Although in this research only PHP applications platforms ware considered unlike experiment in (Djuric, 2013) in which three different applications platform ware used to validate proposed web firewall.

ACCURACY FOR EXPERIMENT 3= TPV/TKV * 100, FPV/TKN *100

Total number of true positive attack (TPA) =3,
Total number of known attack (TKA) =4
Therefore, ACCURACY = 3/4 * (100%) , =75%
Total number of false positive attack (FPA) =1,
Total number of known attack (TKA)=4
Therefore, ACCURACY = 1/4 * (100%) , =25%



4. RESEARCH FINDINGS

This research has used three different applications with different scenarios for preventing SQLIA attack to overcome issue faced by most previous work, as noted early in this research. (See chapter 1 section 1.2) The proposed research woks as intended to introduced pattern matching approach to combat SQL attack by proposing web application firewall as barrier. The research have achieved low false alarm and improve the effectiveness of true number of SQLI Attack as indicated in section 5 (See experiments and analysis)

The result of proposed research in presented in Chapter 5 has show quite improvement with first experiment accuracy 88%, second experiment with 77% and third experiment 79.4% accuracy as presented in Chapter 5 Figure 5.4, Figure 5.7 and Figure 5.10 respectively.

5. RESEARCH CONTRIBUTION

The research work contribute to overcoming challenge of false alarm (false negative and false positive) SQLI attacks by existing web application firewall, which is

common problem in field of SQLI attacks detection and prevention tools.

The contribution aspect of this work is as follows:

I. Improve pattern matching mechanism to recognize and Malicious SQLI attack.
II. Research work improves database of SQL attacks pattern of tautology attack related type to recognize and block any tautology attack.
III. Research work introduced new way of preventing blind SQLI Attack by grouping potential malicious SQLI attack keywords in different database.

6. CONCLUSION

Combating SQLI attacks on Web-based database driving applications required frequent SQL injection vulnerability assessment patches. Applying manual SQL injection vulnerability assessment is required knowledge of how SQL injection vulnerability looks like and how to exploit them. Similarly Manual inspection of SQL injection Attack web firewall is time consuming, costly mostly leave dangerous SQL injection vulnerability undiscovered. In this research an automatic SQL injection web Attack firewall is being proposed to enable SQL injection vulnerability assessment in effective way.
This research in Chapter 1 section two clarified the two major issues of current SQL Web application firewall; fist is low detection of injected SQL attack and reporting high false negative SQL attacks in target application. These issues present a challenge to security administrator during while trying to SQL injection threat. A number of techniques and method has been proposed (See chapter 2) to tackle this challenge however, none of them have completely address this challenge.

Alternately, this research proposed SQL injection web application firewall that applies SQL prevention in dynamic way. This research has been conducted to reduced number of false negative and false positive result of SQL injection web application firewall. To evaluate the proposed SQL injection web application firewall with respect to various literatures studied this research chooses to conduct three different experiment to simulate similar scenarios that related researches have been conducted and accuracy matrices were use to analyze the result.

7. FUTURE WORK

Propose web application firewall faces two major challenges. One is in ability detect new future tautology and blind SQLI attacks that have not been included in propose signature pattern. With time constraint the study could not able to develop pattern that will able to learn from future attacks of same type considered.
Another challenge faced by proposed web application firewall is reporting of similar attacks more than one times which result in false positive alarm. This is one of the reasons why proposed firewall report average of 27% of false positive alarm.
In short the future research work to address on this work is as follows:
i. Improving firewall pattern matching
ii. Updating database of blind and tautology SQLI attacks signatures.
iii. Reducing number of false negative alarm by introducing two ways detection approach.

REFERENCES

[1] Acunetix. (2013). Accunetix Vulnerability Scanner. Retrieved 29/06/2015, from https://www.acunetix.com/vulnerability-scanner/
[2] Agosta, Giovanni, Barenghi, Alessandro, Parata, Antonio, & Pelosi, Gerardo. (2012). Automated security analysis of dynamic web applications through symbolic code execution. Paper presented at the Information Technology: New Generations (ITNG), 2012 Ninth International Conference on.
[3] Antunes, Nuno, & Vieira, Marco. (2009a). Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services. Paper presented at the Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on.
[4] Antunes, Nuno, & Vieira, Marco. (2009b). Detecting SQL injection vulnerabilities in web services. Paper presented at the Dependable

Computing, 2009. LADC'09. Fourth Latin-American Symposium on.

[5] Antunes, Nuno, & Vieira, Marco. (2010). Benchmarking vulnerability detection tools for web services. Paper presented at the Web Services (ICWS), 2010 IEEE International Conference on.

[6] Antunes, Nuno, & Vieira, Marco. (2011). Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services. Paper presented at the Services Computing (SCC), 2011 IEEE International Conference on.

[7] Antunes, Nuno, & Vieira, Marco. (2012). Evaluating and improving penetration testing in web services. Paper presented at the Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on.

[8] Antunes, Nuno, & Vieira, Marco. (2015). Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples. Services Computing, IEEE Transactions on, 8(2), 269-283.

[9] Appelt, Dennis, Nguyen, Cu Duy, Briand, Lionel C, & Alshahwan, Nadia. (2014). Automated testing for SQL injection vulnerabilities: an input mutation approach. Paper presented at the Proceedings of the 2014 International Symposium on Software Testing and Analysis.

[10] Bandhakavi, Sruthi, Bisht, Prithvi, Madhusudan, P, & Venkatakrishnan, VN. (2007). CANDID: preventing sql injection attacks using dynamic candidate evaluations. Paper presented at the Proceedings of the 14th ACM conference on Computer and communications security.

[11] Bau, Jason, Bursztein, Elie, Gupta, Divij, & Mitchell, John. (2010). State of the art: Automated black-box web application vulnerability testing. Paper presented at the Security and Privacy (SP), 2010 IEEE Symposium on.

[12] Boyd, Stephen W, & Keromytis, Angelos D. (2004). SQLrand: Preventing SQL injection attacks. Paper presented at the Applied Cryptography and Network Security.

[13] Buehrer, Gregory, Weide, Bruce W, & Sivilotti, Paolo AG. (2005). Using parse tree validation to prevent SQL injection attacks. Paper presented at the Proceedings of the 5th international workshop on Software engineering and middleware.

[14] Cenzic's. (2014). Application Vulnerability Trends Report: 2014. Retrieved 29/09/2015, from https://www.info-point-security.com/sites/default/files/cenzic-vulnerability-report-2014.pdf

[15] Chen, Jan-Min, & Wu, Chia-Lun. (2010). An automated vulnerability scanner for injection attack based on injection point. Paper presented at the Computer Symposium (ICS), 2010 International.

[16] Cheon, Eun Hong, Huang, Zhongyue, & Lee, Yon Sik. (2013). Preventing SQL Injection Attack Based on Machine Learning. International Journal of Advancements in Computing Technology, 5(9).

[17] Cho, Ying-Chiang, & Pan, Jen-Yi. (2015). Design and Implementation of Website Information Disclosure Assessment System. PloS one, 10(3), e0117180.

[18] Ciampa, Angelo, Visaggio, Corrado Aaron, & Di Penta, Massimiliano. (2010). A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. Paper presented at the Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems.

[19] Cook, William R, & Rai, Siddhartha. (2005). Safe query objects: statically typed objects as remotely executable queries. Paper presented at the Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on.

[20] Djuric, Zoran. (2013). A black-box testing tool for detecting SQL injection vulnerabilities. Paper presented at the Informatics and Applications

(ICIA), 2013 Second International Conference on.

[21] Falcove. (2007). Falcove Web Vulnerability Scanner and Penetration Testing. Retrieved 29/06/2015, from http://www.ramsayfalcove.com/htdocs/Welcome.html

[22] Fu, Xiang, Lu, Xin, Peltsverger, Boris, Chen, Shijun, Qian, Kai, & Tao, Lixin. (2007). A static analysis framework for detecting SQL injection vulnerabilities. Paper presented at the Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International.

[23] Gartner. (June 19, 2014). WEB APPLICATION ATTACK REPORT #5. Retrieved 29/06/2015, from http://www.imperva.com/docs/hii_web_application_attack_report_ed5.pdf

[24] Grendel.). Grendel-Del Web vulnerability Scanner. Retrieved 29/06/2015, from http://sectools.org/tool/grendel-scan/

[25] Halfond, William GJ, & Orso, Alessandro. (2007). Detection and prevention of sql injection attacks Malware Detection (pp. 85-109): Springer.

[26] Huang, Shih-Kun, Lu, Han-Lin, Leong, Wai-Meng, & Liu, Huan. (2013). Craxweb: Automatic web application testing and attack generation. Paper presented at the Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on.

[27] Huang, Yao-Wen, Tsai, Chung-Hung, Lin, Tsung-Po, Huang, Shih-Kun, Lee, DT, & Kuo, Sy-Yen. (2005). A testing framework for Web application security assessment. Computer Networks, 48(5), 739-761.

[28] Huang, Yao-Wen, Yu, Fang, Hang, Christian, Tsai, Chung-Hung, Lee, Der-Tsai, & Kuo, Sy-Yen. (2004). Securing web application code by static analysis and runtime protection. Paper presented at the Proceedings of the 13th international conference on World Wide Web.

[29] IBM. (2013). IBM Web Application Scanner. Retrieved 29/06/2015, from http://www-03.ibm.com/software/products/en/appscan

[30] Inspect, HP. (2012). HP Inspect Vulnerability. Retrieved 29/06/2015, from http://sectools.org/tool/webinspect/

[31] Jnena, Rami MF. (2013). Modern Approach for WEB Applications Vulnerability Analysis. The Islamic University of Gaza.

[32] Joshi, Anamika, & Geetha, V. (2014). SQL Injection detection using machine learning. Paper presented at the Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on.

[33] Kals, Stefan, Kirda, Engin, Kruegel, Christopher, & Jovanovic, Nenad. (2006). Secubat: a web vulnerability scanner. Paper presented at the Proceedings of the 15th international conference on World Wide Web.

[34] Khoury, Nidal, Zavarsky, Pavol, Lindskog, Dale, & Ruhl, Ron. (2011). An analysis of black-box web application security scanners against stored SQL injection. Paper presented at the Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on.

[35] Kumar, Praveen. (2013). The multi-tier architecture for developing secure website with detection and prevention of sql-injection attacks. International Journal of Computer Applications, 62(9), 30-35.

[36] Lawal, MA, Sultan, Abu Bakar Md, & Shakiru, Ayanloye O. (2016). Systematic Literature Review on SQL Injection Attack. International Journal of Soft Computing, 11(1), 26-35.

[37] Liban, Abdilahi, & Hilles, Shadi. (2014). Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack. Paper presented at the Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th.

[38] Liu, Anyi, Yuan, Yi, Wijesekera, Duminda, & Stavrou, Angelos. (2009). SQLProb: a proxy-based architecture towards preventing SQL injection attacks. Paper presented at the Proceedings of the 2009 ACM symposium on Applied Computing.

[39] Livshits, V Benjamin, & Lam, Monica S. (2005). Finding Security Vulnerabilities in Java Applications with Static Analysis. Paper presented at the Usenix Security.

[40] McClure, Russell A, & Kruger, Ingolf H. (2005). SQL DOM: compile time checking of dynamic SQL statements. Paper presented at the Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on.

[41] Medhane, Munqath H Alattar SP. R-WASP: Real Time-Web Application SQL Injection Detector and Preventer.

[42] N-Stalker. (27 Feb. 2014,). N-Stalker Web Vulerability Scanner. Retrieved 29/06/2015, from http://www.windowsecurity.com/software/Web-Application-Security/N-Stalker-Web-Application-Security-Scanner.html

[43] Nguyen-Tuong, Anh, Guarnieri, Salvatore, Greene, Doug, Shirley, Jeff, & Evans, David. (2005). Automatically hardening web applications using precise tainting: Springer.

[44] OWSAP. (2013). Top 10 Vulnerability 2013 by Open Web Security Project. Retrieved 29/06/2015, from https://www.owasp.org/index.php/Top_10_2013-Top_10

[45] Qu, Binbin, Liang, Beihai, Jiang, Sheng, & Chutian, Ye. (2013). Design of automatic vulnerability detection system for Web application program. Paper presented at the Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on.

[46] Scott, David, & Sharp, Richard. (2002). Abstracting application-level web security. Paper presented at the Proceedings of the 11th international conference on World Wide Web.

[47] Shahriar, Hossain, & Zulkernine, Mohammad. (2012). Information-theoretic detection of sql injection attacks. Paper presented at the High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on.

[48] Shar, Lwin Khin, & Tan, Hee Beng Kuan. (2012). Predicting common web application vulnerabilities from input validation and sanitization code patterns. Paper presented at the Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on.

[49] Shin, Yonghee, Williams, Laurie, & Xie, Tao. (2006). Sqlunitgen: Sql injection testing using static and dynamic analysis. Paper presented at the 17th IEEE International Conference on Software Reliability Engineering, ISSRE.

[50] Singh, Avinash Kumar, & Roy, Sangita. (2012). A network based vulnerability scanner for detecting SQLI attacks in web applications. Paper presented at the Recent Advances in Information Technology (RAIT), 2012 1st International Conference on.

[51] Thiyagarajan, A, Uma, S, Vipin, Ambat, & Dheen, Najeem. (2015). METHODS FOR DETECTION AND PREVENTION OF SQL ATTACKS IN ANALYSIS OF WEB FIELD DATA.

[52] Tiwari, Yash, & Tiwari, Mallika. (2015). A Study of SQL of Injections Techniques and their Prevention Methods. International Journal of Computer Applications, 114(17).

[53] Valeur, Fredrik, Mutz, Darren, & Vigna, Giovanni. (2005). A learning-based approach to the detection of SQL attacks Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 123-140): Springer.

[54] Zhang, Xin-hua, & Wang, Zhi-jian. (2010). Notice of Retraction A Static Analysis Tool for Detecting Web Application Injection Vulnerabilities for ASP Program. Paper presented at the e-Business and Information System Security (EBISS), 2010 2nd International Conference on.