Efficient Solution for Detection and Prevention of SQL Injection Attacks (Wave system technique)

Mona Medhat
Teaching Assistant at Management Information Systems Department, Higher Institute For Specific Studies, Cairo, Egypt

Christina Albert
Assistant Professor at Computer and Information System Department, Sadat Academy for Management Science, Cairo, Egypt

Mohamed M. EL HADI
Professor at Computer and Information Systems Department, Sadat Academy for Management Science, Cairo, Egypt

Abstract

SQL Injection attacks are one of the most common threats on web applications that refer to an attacker who can use vulnerability to bypass authentication for retrieving the contents of an entire database then create, delete, update or drop the whole structure. There are many methods used to repel these attacks but none of these methods have proved to work on detecting and preventing all types of SQL injection attacks which means specific method for a certain particular type. The aim of this research is to present a new method to detect and prevent the largest number of these attacks and test it against the 50 codes written by PHP and HTML languages Analysis and comparison have been carried out between the existing solutions YASCA, RIPS and WAVE, questionnaires were completed by experts such as developers and database administrators and identification of the actual risks behind these threats have all helped in addressing the best method to use in securing websites.

Keywords: SQL Injection, Attack, Prevention-Detection, vulnerability attack. Threats, SQLIA..

1. Introduction

Modern era is totally dependent on technology and one of technology's most important aspects that affect the individual's day to day life is web applications and their data bases. Therefore securing the transaction done through these web applications is extremely important up till now the security level is considered weak and the main weakness is related to the process of identifying the user. An unauthorized user can easily access the system by via SQL injection input which is considered the weak point in the security of the web applications thus we are going to try and shed the light through survey Of the vulnerabilities of the attacks on the different types of SQL injection attacks and also on the techniques of their prevention and how to detect them in the first place. We also have to mention the new kind of attack called WAVE system tool.

Section 2 in this paper presents the types of SQL injection attacks and techniques to prevent those attacks from hacking, section 3 shows the steps involved in the process of static slicing, section 4 shows Architecture and the experimental testing wave system, vulnerabilities detected by The WAVE System, RIPS tool and YASCA tool also shows the result of the questionnaire and survey, section 5 shows the Result, section 6 show the conclusion, section 7 shows the future work.

2. Types of SQL injection attacks

There are different methods of attacks that depend on the goal of the attacker. For a successful SQLIA the attacker should append on a syntactically correct command to the original SQL query and types of SQL as follows: -

• Tautologies: This type of attack injects SQL tokens to the conditional query statement to be evaluated always true (Halfond, Viegas, &Ors 2006)

• Illegal/Logically Incorrect Queries: When a query is rejected, an error message is returned from the database including useful debugging information. This error message helps the attacker to find vulnerable parameters in the application and consequently database of the application (Su &Wassermann, 2006).

• Union Query: In this type of attack, intruders exploit database by the query delimiter such as ";" to append extra query to the original query. With a successful attack, database receives and executes a multiple distinct queries. Normally the first query is a legitimate query, whereas the following queries could be illegitimate (Halfond, Viegas, &Orso, 2006)

• Stored Procedure: In this Technique, attacker focuses on the stored procedures which are present in the database system .Stord procedures run directly by the database engine. stored procedures is nothing but a code and it can be vulnerable as program code for authorized and unauthorized user the stored procedure return true/false

• Alternate Encodings: In this technique attackers modify the injection query by using alternate encoding such as hexadecimal, ASCII, and Unicode. Because by this way they can escape from developer's filter which scan input queries for special known "bad character".

• Inference: By this type of attack intruders change the behaviour of a database or application. There are two well-known attack techniques that are based on inference blind-injection and timing attacks (Motamedi, & Akbari, 2009)

• Blind Injection: Sometimes developers hide the error details which help attackers to compromise the database. So the SQLIA would be more difficult but not impossible. An attacker can still steal data by asking a series of true/false questions through SQL statements.

• Timing Attacks: A timing attack lets the attacker gathers the information from the database by observing timing delays in the database's responses. This technique uses an if-then statement for injecting queries. WAITFOR is a keyword along the branches which causes the database to delay its response by a specified time.

3. Techniques for prevention and Detection website Application from SQL Injection Attacks

Researchers have proposed a wide range of techniques to address the problem of SQL injection. These techniques range from developing best practices to fully automated tools for detecting and preventing SQLIAs. In this section, these proposed tools will be reviewed, summarized and the advantages and disadvantages associated with each tool will be highlighted (Sharifian, Motamedi, & Akbari, 2009). It is noticeable that there are more techniques that have not yet been implemented as a tool.

• In SQL Guard and SQL Check queries are checked at runtime based on a model which is expressed as a grammar that only accepts legal queries. SQL Guard examines the structure of the query before and after the addition of user-input based on the model. In SQL Check the model is specified independently by the developer. Both approaches use a secret key to delimit user input during parsing by the runtime checker, thus security of the approach is dependent on attackers not being able to discover the key. In both approaches developers are be able to modify the code to use a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generated query.

• Signature Approach: Signature Approach as it appears from its name is based on the signature thus the injection is caused through validating the input. This approach uses Hirschberg algorithm to check the statements from specifications versus the one from a hotspot. There are three modules to do so and they are as follows:-

Monitoring module: This depends on the relation of the input taken from web application sent to the module for matching and in case of mismatch an error message is sent and the transaction is blocked.

Specification predefined: key words are stored in the database against which we check the input (Xi-Rong Wu; &Chan, P.P.K., 2012).

Analysis Module:-It uses Hirschberg algorithm to compare an input taken from monitoring Module with a hotspot.

• DIWEDA Approach as well as ROICHMAN is mainly a framework to identify SQL injection and business logic violation and that is through IDS (Intrusions detection system) for backend database. Therefore DIWEDA is considered a prototype that works at the session level and not SQL statement.

• SAFELL: is static analysis framework that aims to detect the vulnerability of the SQL analysis and its target is to identify the SQL injection attack at the compile time. SAFELL has two assets the first one is the analysis of black box testing which takes into consideration the byte code and track string. Second one deals with Boolean integer and string variable because it depends on hybrid constraint solver that is considered a strong string analysis tool.

• Black box testing: protects against SQL injection attacks through machine learning techniques (Dharam, R.; Shiva, S.G., 2012). This is done by defining the points that are considered vulnerable to attacks and that is through web crawler. However the main disadvantage is that is does not secure total protection.

• JBBC-checker: This is considered a limited technique because it only covers tautologies and cannot protect other kinds of attack. In other words it detects the mismatches injection and also protects against them but it is limited to that.

• Combined static and dynamic analysis AMNESIA: is technique that combines dynamic and static analysis for detecting and preventing web application vulnerabilities at run time .AMNESIA uses static analysis to generate different types of query statement. In the dynamic phase AMNESA interprets all queries before they are sent to the database and validates each query against the statically built model. AMNESA stops al queries before they are sent the database and validates each query statement against the AMNESA models

Analysis module: this kind of modules have java applications as input and a group of hotspots as an output having a SQL query for each on of then. This is done in two steps:-Instrumentation module: it is the next technique in line where the java web application is the input and the group of hotspots are linked to the runtime monitors. Runtime monitoring module: this module comes next. It recalls the SQL Query model for a hotspot and compares it with the query model. This is done after taking a query string and ID of the hotspot as an input (Scott & Sharp, 2003).

• Identify hotspots: this is done by identifying through scanning and thus declares the application code as issue SQL queries to the underlying database.

• Build SQL query models this depends on building a model that carries all the SQL queries that would possibly be produced at certain hotspots.

• SQL query model: is automated non deterministic finite state. The transaction is a label that is composed of SQL tokens. Delimiters and place holders for string values.

• Limitation: the result of these techniques is either false positive and false negative thus its forte relies on the built model (Wassermann, Gould & Su, 2004).

• SQL DOM: Framework that is represented by McClure and Kruge, they estimate from the existing flow while accessing relation database from object oriented programming languages. They focus on with the database SQL DOM object model is design to take these through building a secure environment for communication.

• SQL RAND (Randomization based method): we can express "randomize SQL as parts of the query generated by an application and use correctly randomize in SQL to detect attacks. SQL RAND creates instance of language that are unpredictable to the attacker. One that the attacker can't easily guess we define de-randomization proxy which converts randomized query to proper SQL query for database. Our design consist of a proxy that sits between the client and database server proxy may be on a separate machine by moving de-randomization process outside DBMS to the proxy we gain that flexibility, simplicity and security.

• WebSSARI: uses static analysis to check taint flows against preconditions for sensitive functions. It works based on sanitized input that has passed through a predefined set of filters. The limitation of this approach is adequate preconditions for sensitive functions cannot be accurately expressed so some filters may be omitted (Bandhakavi, Bisht, & Madhusudan, 2007).

• SecuriFly: is another tool that was implemented for java. Despite other tools chasing string instead of character for taint information, Security Fly tries to sanitize query strings that have been generated using tainted input but unfortunately injection in numeric fields cannot be stopped by this approach. Because difficulty of identifying all sources of user input is the main limitations of this approach (Martin, Livshits, and Lam, 2009).

• Positive tainting: not only focuses on positive tainting rather than negative tainting but also it is automated and does not need developer intervention. Moreover this approach benefits from syntax-aware evaluation which gives developers a mechanism to regulate the usage of string data based not only on its source but also on its syntactical role in a query string.

• Intrusion Detection System (IDS) is based on a machine learning technique. That is trained using a set of typical application queries. the technique builds models of the typical queries and then monitoring the application at runtime to identify queries that do not match the model. In their evaluation, valeure and colleagues have shown that their system is able to detect attacks with a high rate of success however; the fundamental limitation of learning based techniques is that they can provide no guarantees about their detection abilities because their success is dependent on the quality if the training set used. A poor training set would cause the quality of the learning technique to generate a large number of false positive and negatives.

• Swaddler analysis: the internal state of a web application. It works based on both single and multiple variables and shows an impressive way against complex attacks to web applications. First the approach describes the normal values for the application's state variables in critical points of the application's components. Then, during the detection phase it monitors the application's execution to identify abnormal states (Cova, M, Balzarotti, D., 2012).

• RIPS: is a tool written in PHP to find vulnerabilities in PHP applications, using static code analysis by tokenizing and parsing all source code files. RIPS has the ability to transform PHP source code into program flow.

• Wave system techniques (web application vulnerability Extractor)

• Algorithm for Detecting and preventing website Application from SQL injection attacks

An efferent algorithm for detecting and preventing SQL Injection Attacks is based on wave system .the planned architecture is given in fig(1) The main wave system of architecture is the predictive parser which is the stack, that manipulates the tokens according to the information in the predicative context free grammar table and parsing table. The parser utilizes the stack to store the production rule associated with the current token. This rule will serve to define the appropriate production rule for the next token. Given a new token, the parser check the parsing table for the appropriate production rule according to the non-terminal exist on the stack top. Accordingly, the stack will be updated with the new production rules after popping the non-terminal from it. The predicative context free grammar is a list containing the productions rules of the PHP web programming language. The parsing table is created from the predicative context free grammar. It stores the actions that the parser should take based on the input token and the value on the stack top.

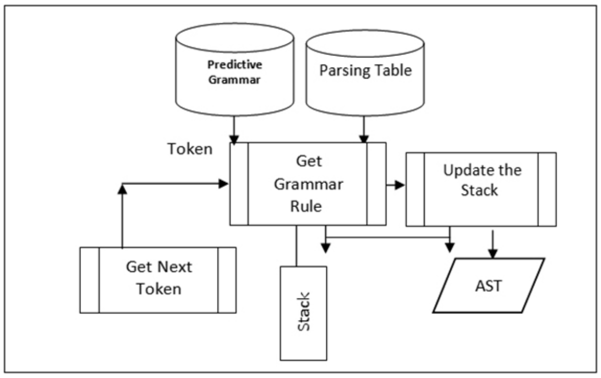


Figure 1: wave system Architecture

o The procedure of proposed model (wave) system

A) Static Slicing:

Static slicing is the approach to analyse the code statistically to isolate specific statements. In general, static slicing process isolates the statements which attain vulnerabilities directly or indirectly. The process extracts a partition of the program or a set of nodes that directly or indirectly affect the value of the set of variables{V} at a specific point in the program 'S'. The specific point is known as the cut point, whereas, the extracted portion is called a slice. The slicing criterion is denoted by (S, { V}) where 'S' is a statement, node or line number and {V} is the set of program variables under study [Surendran et al., 2013].

Detecting web security input vulnerabilities requires checking the entire code of the

Application. Therefore 'S' will represent the last statement of the code and 'V' is the set of input variables and their influenced variables. Direct input variables are those whose values are obtained from the user (user inputs). The direct input are, normally, hold in special arrays called $_GET, $_POST, $_REQUEST and $_COOKIE. The influenced variables are those whose values were defined (directly or indirectly) by direct input, and those whose execution are controlled (directly or indirectly) by any variable in {V}. The slicer manipulates the generated define-use chain in the form of linked lists as well as the decision nodes linked list to extract the code slice. The static slice of the code is presented as linked list. Whereas the Constrained Slicer optimizes the extracted slice to obtain the slice of the slice which is a set of vulnerable instructions (code statements). Figure.2 shows the steps involved in the process of static slicing.

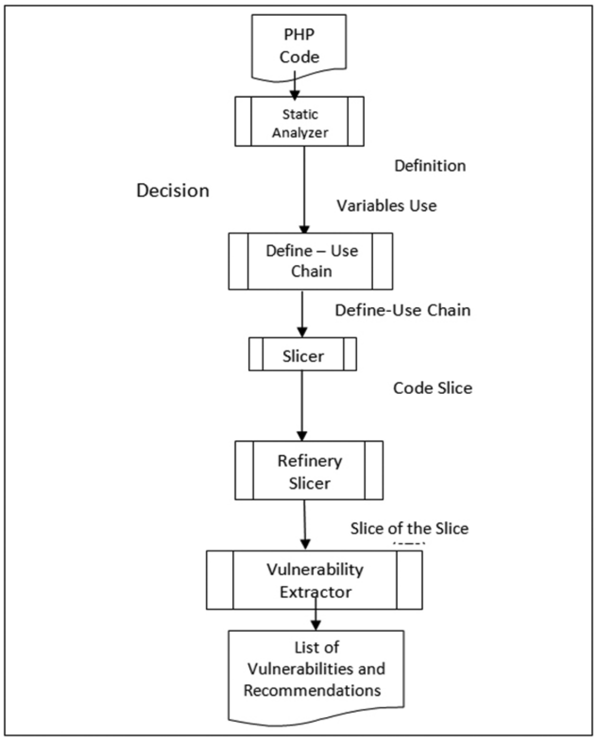


Figure 2: steps involved in the process of static slicing.

B) Defined-Use Chain Table

Defined-Use chain of a given variable v, describes the list of definition nodes of v, and the associated use nodes of the corresponding reaching definitions.

The Defined-Use chain is represented in a simple linked list data structure as shown in Figure 3.

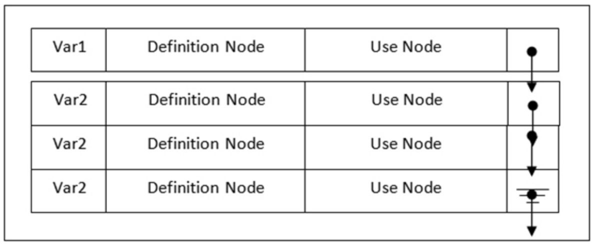


Figure.3: Data Structure for Defined-Use Chain

c) Defined-Use Chain Extraction

The define-use chain extractor function uses the definition table and use table in order to generate the define-use chain. (M. Salah, 2014)

For the purpose of obtaining the data influence of the input variable(s), the code should be analyzed up to its last statement. Therefore the slicing node is selected to be the last node of the program. The best structure for storing the file slice is a simple data linked list. 4. Extracting the Slice from Define-Use Chain

The function that is reasonable for creating the code slice is called Slicer. The Slicer function initializes the slice with those nodes that contains external data variables (data comes from the user) Afterword it delivers the slice by repeating the following steps until saturation.

• Add to the slice the definition nodes of any variable that has been used in the slice.

• Add to the slice the decision nodes that any of the statements in the slice is in its scope.

D) Slicing the Slice

Slicing the slice (STS) is second iteration which made over the slice obtained by applying static slicing. STS extracts, only, the statements that contain direct or indirect input variables by applying data influence over input variables.

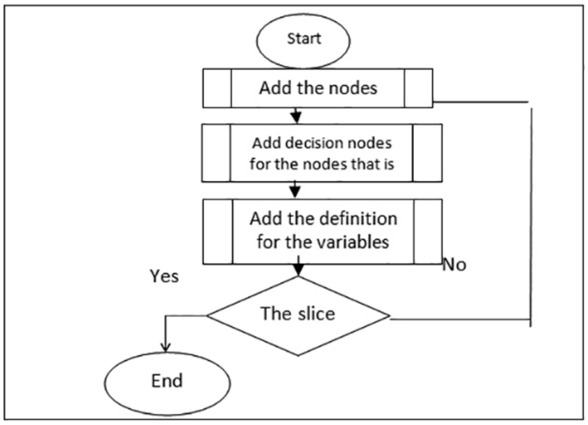

Start
Add the nodes
Add decision nodes for the nodes that is
Add the definition for the variables
No
Yes
The slice
End

Figure 4: Block Diagram for the Slicer Processes

STS is the set of nodes that will reflect any data vulnerabilities that should need remedies recommendations.

e) Extracting the STS from the PHP Code

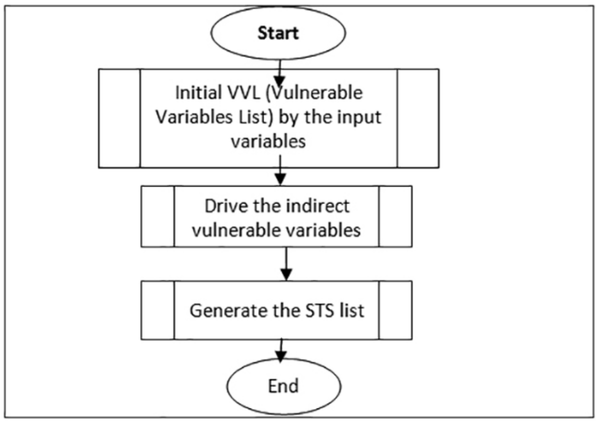

Start
Initial VVL (Vulnerable Variables List) by the input variables
Drive the indirect vulnerable variables
Generate the STS list
End

Figure 5: Block Diagram for the STS Processes

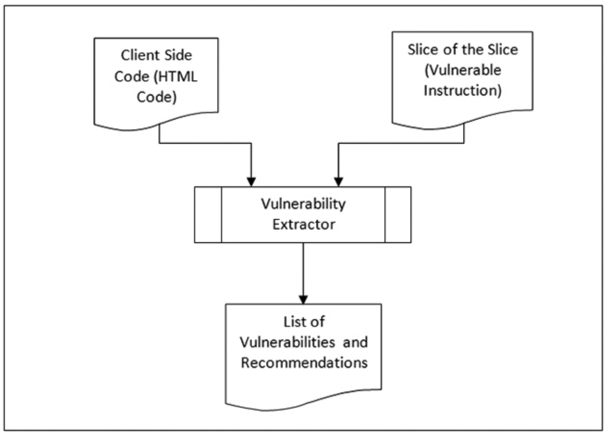

Client Side Code (HTML Code)
Slice of the Slice (Vulnerable Instruction)
Vulnerability Extractor
List of Vulnerabilities and Recommendations

Figure 6: Block Diagram for Vulnerability Extractor

f) Vulnerability Extraction

This section discusses the detection of the vulnerabilities using the proposed system. The analysis of the PHP code to extract its vulnerabilities and provides some remedies using the proposed system will be shown. The resolved code will be exercised with real time attack to obtain the effect of the diagnosed deficiencies and its remedies. Figure 6: shows the block diagram of the vulnerability extractor, which manipulates the client side code and the derived slice of the slice (STS) as inputs and analyze them to extract a list of vulnerabilities and recommendations.(M. Salah, 2014)

The vulnerabilities of the PHP code will be extracted and a remedy for it will be recommended according to each vulnerability type. This will be accomplished by exercising the code through the proposed WAVE system, which isolates the statements that causes its vulnerabilities. The code will processed to obtain the static slice, which, is refined more to obtain the precise vulnerability list. A recommendation remedy is provided to mend the statement in the vulnerability list. In the next subsections, the technique of detecting each of the web security vulnerabilities will be introduced.

4- Results from testing Evaluation

4.1. Testing the WAVE System

The WAVE system was exercised by running it with the prepared test suite. It detects 80 vulnerabilities as shown in Table 1 and Figure 3 the entries in the table signify the number of vulnerabilities detected and extracted by the WAVE system. The columns presents the type of vulnerability, while, the rows show the category of the tested code.

	Buffer overflow	Cookie Poisoning	SQL Injection	Cross Site Script	Bypass input restrictions	CGI Parameters Attack	Hidden field attack
Papers Codes	2	1	3	6	0	6	0
Websites Codes	6	6	2	8	7	8	0
Synthetic Codes	4	1	4	4	1	8	1
Total	14	8	9	18	8	22	1

Table 1: The Vulnerabilities Detected By the WAVE System

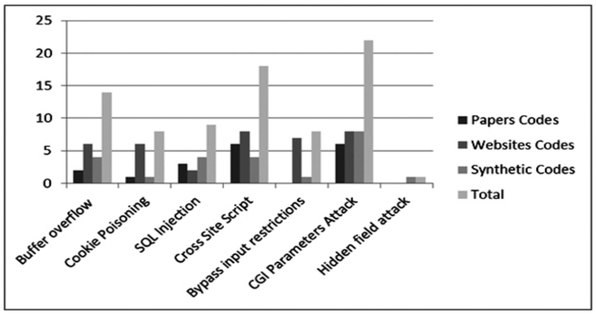


Figure 7: The Vulnerabilities Detected By the WAVE System

type of attacks	WAVE	YASCA	RIPS
sql injection attacks	9	2	4
Cross site script	18	4	0
cookie poisoning attack	8	0	0
Bypass Restriction	8	0	0
CGI parameter	22	0	0
Buffer Over flow	14	0	0
Hidden	1	0	0
total	80	6	4

Table 2: Vulnerabilities Detected by WAVE System, RIPS Tool and YASCA Tool

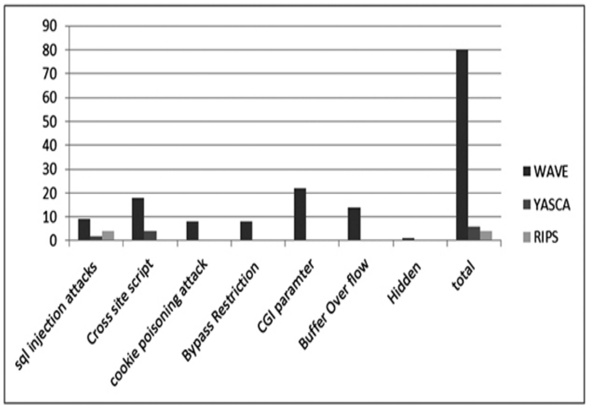
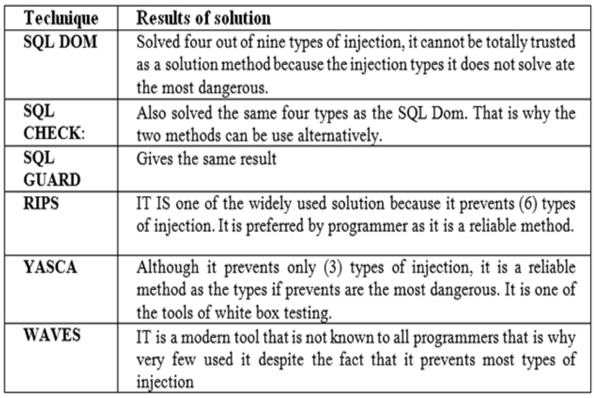


Figure 8: Vulnerabilities Detected By WAVE System, RIPS Tool and YASCA Tools

4.2. Questionnaire Design and Evaluation

This study is both an analytical and afield study. The analytical part aims at choosing the best existing method to prevent hacking on web application. The choice of the suitable preventing method depends on: security, securing code and time management. The field study depends on a number of questions concerning using SQL and protecting web application against SQL injection. The questions were answered by (81) programmers, web developer, senior web developer and DB admin. (100 questionnaires were handed out). Their answers were analyzed and the result and recommendation as seen from the table:



Technique	Results of solution
SQL DOM	Solved four out of nine types of injection, it cannot be totally trusted as a solution method because the injection types it does not solve ate the most dangerous.
SQL CHECK:	Also solved the same four types as the SQL Dom. That is why the two methods can be use alternatively.
SQL GUARD	Gives the same result
RIPS	IT IS one of the widely used solution because it prevents (6) types of injection. It is preferred by programmer as it is a reliable method.
YASCA	Although it prevents only (3) types of injection, it is a reliable method as the types if prevents are the most dangerous. It is one of the tools of white box testing.
WAVES	IT is a modern tool that is not known to all programmers that is why very few used it despite the fact that it prevents most types of injection

Table 3: analyzed and the result and recommendation

5. Conclusion

It is difficult to objectively create test codes for the static analysis tools because there is no widely agreed-upon yardstick for comparing results. The most common method to evaluate a static analysis tool is to run it on a series of real-world applications and manually inspect the results.

A test suite (set of PHP codes) has been used to evaluate the results of the designed WAVE system. It consists of codes varying in size and complexity. The selection of the test codes were made without any bias towards a specific static analysis tool. The WAVE system was exercised by running it with the prepared test suite. It detects 87 vulnerabilities.YASCA tool can't detect the web vulnerabilities in code that contain indirect input variable. Both RIPS and YASCA tools ignored the cross site script vulnerability, which occurs due to outputting to a database. They also ignored the SQL injection vulnerability when vulnerable SQL statement did not pass to the database in the same PHP page. This section is a comparison of the RIPS, YASCA, and WAVES tools as far as the ability of each of them to discover and prevent different vulnerabilities (name of injection) is concerned. Each tool is applied on 50 injected codes. 28 codes are obtained from websites, 11 codes are written for testing purpose and 10 are taken from different research papers. They are all complex codes, which help in testing and evaluating how successful the tool is in detecting and preventing vulnerabilities. The most common method to evaluate a static analysis tools is to run it on a series of real world application and manually inspect the result. In addition, the section explains the way each tool analyses the codes showing their strong and weak points. Accordingly,

it achieves some recommendation to clarify the best tool to prevent websites from different vulnerabilities. In this chapter, 50 codes are examined by three methods WAVE, RIPS and YASCA the following was observed. The YASCA method has merely detected 6 code injections from a total of 50 codes two of these 6 codes were SQL injection and 4 of them were cross- site script. The RIPS method has only detected 4 codes for total of 50 codes. Indeed the four of them were SQL injection. The WAVE method has detected 80 there was a wide range of vulnerabilities such as cross-site script, buffer over flow, SQL injection, legal query and stored procedure. When it comes to the YASCA method, the YASCA tool is focused on direct input variable, regular expression, and patterns. So it can detect vulnerabilities related to these methods. However, a major weak point in the YASCA is that it does not follow the path of the variable. This can cause a problem to the server because the variable can inject it.

6. Future work

The WAVE system could be extended to support other web programming language such as ASP and JSP. This could be accomplished by generating data influence tables from the parse tree of their front-end compilers. The WAVE system could, also, be adapted to support multiple pages and object oriented programming. While, the presented WAVE system adopted a static approach (static slicing technique) to expose the vulnerabilities of PHP code, a dynamic approach for detection of vulnerabilities is an open research area, dynamic analysis techniques could play a great role in security assurance.

7-References

1-Bandhakavi, S, Bisht, P., & Madhusudan, P (2007) CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations, USA, ACM.

2- Halfond, W., Viegas. J & Orso, A.(2006) "A Classification of SQL Injection Attacks and Countermeasures," College of Computing Georgia Institute of Technology IEEE.

3- Milne, G. R., L. I. Labrecque & C. Cromer.(2009.) "Toward an understanding of the online consumer's risky behaviour and protection practices." Journal of Consumer Affairs 43(3): 449-473.

4- Scott, D and Sharp, R., (2003) Abstracting Application-level Web Security. IEEE---Transactions on Knowledge and Data Engineering (Volume: 15, Issue: 4)

5- Wassermann, G; Gould, C; Su, Z, et al, (2004)" Static Checking of Dynamically Generated Queries in Database Applications,"(Volume: 16, Issue) ACM Transactions on Software Engineering and Methodology.

6-[Surrendering, 2013] Surrendering, A.; Samuel, P. and Poulose, K."Code Clones in Program Test Sequence Identification", International Journal of Computer Information Systems and Industrial Management Applications, Pages 564-570, India, 2013.

7--Halfond, W. G. J. & Orso, A. (2006) "Preventing SQL injection attacks using AMNESIA," presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China.

8-Martin, M. Livshits, B. & Lam, M (2009) Finding Application Errors and Security Flaws Using PQL (Vol: 40, Iss: 10, pp: 365-383): A Program Query Language,"ACM SIGPLAN Notices.

9-Sharifian,S., Motamedi, S., & Akbari, M.,(Eds.).(2009). Estimation-Based Load-Balancing with Admission Control for Cluster Web Servers," ETRI Journal, vol.31, pp.173-181.

10-Su .Z., &. Wassermann, G. (2006) "The Essence of Command Injection Attacks in Web Applications". (Volume: 41, pp: 372-382) ACM SIGPLAN Notices.

11- Salah, M. (2014) Verification of Web Applications Vulnerabilities, Institute of Statistical Studies & Research Cairo University, Egypt.

12- Cova, M, Balzarotti, D., (2012) "Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications"(Volume: 4637, pp: 63-Recent Advances in Intrusion Detection, Proceedings.

13-Xi-Rong Wu; Chan, P.P.K.,(2012 July 17) "SQL injection attacks detection in adversarial environments by k-centers," Machine Learning and Cybernetics,(ICMLC), International Conference on , vol.1, no., pp.406, 410.

14-Dharam, R.; Shiva, S.G.,(2012) "Runtime monitors for tautology based SQL injection attacks," Cyber Security, Cyber Warfare and Digital Forensic (Cyber Sec) International Conference on , vol., no., pp.253,258, 26-28 June .