# Identification of Susceptible Websites from Code Injection Attach

Hussein Alnabulsi, Rafiqul Islam

*School of Computing and Mathematics, Charles Sturt University, Albury,2640, Australia*
*School of Computing and Mathematics, Charles Sturt University, Albury,2640, Australia*
*{halnabulsi,mislam}@csu.edu.au*

**Abstract**

The paper presents a framework methodology for identifying the real website is vulnerable to code injection attack, our proposed methodology gives a solution for thousands of websites to identify their vulnerability against code injection attacks, so it can help the administrator of the website by checking his own website and knows if it is vulnerable to code injection attacks. There are not too many researchers about this subject because most of the cyber securityresearchers are about detecting and protecting against the malware, the good thing in this research is that it can provide a self-checking protection for every website, so the administrator can know if his website needs to get more protection support against malware or if the anti-malware that the administrator is using for protection against code injection attack is enough.
The framework that presented in this paper can do a code injection attacks such as SQL injection and XSS attacks by itself to check whether the website is vulnerable to code injection attack. The checking methodology is contained all ways that the attacker can do to success his code injection attack, so it gives a precision checking for the website whether is vulnerable against code injection attack or no. By this way, we can protect many of websites against hackers that using code injection attack to make damage such as stealing or deleting or altering important data from website systems.

*Keywords*: code injection attack, XSS attack, SQL injection attack;

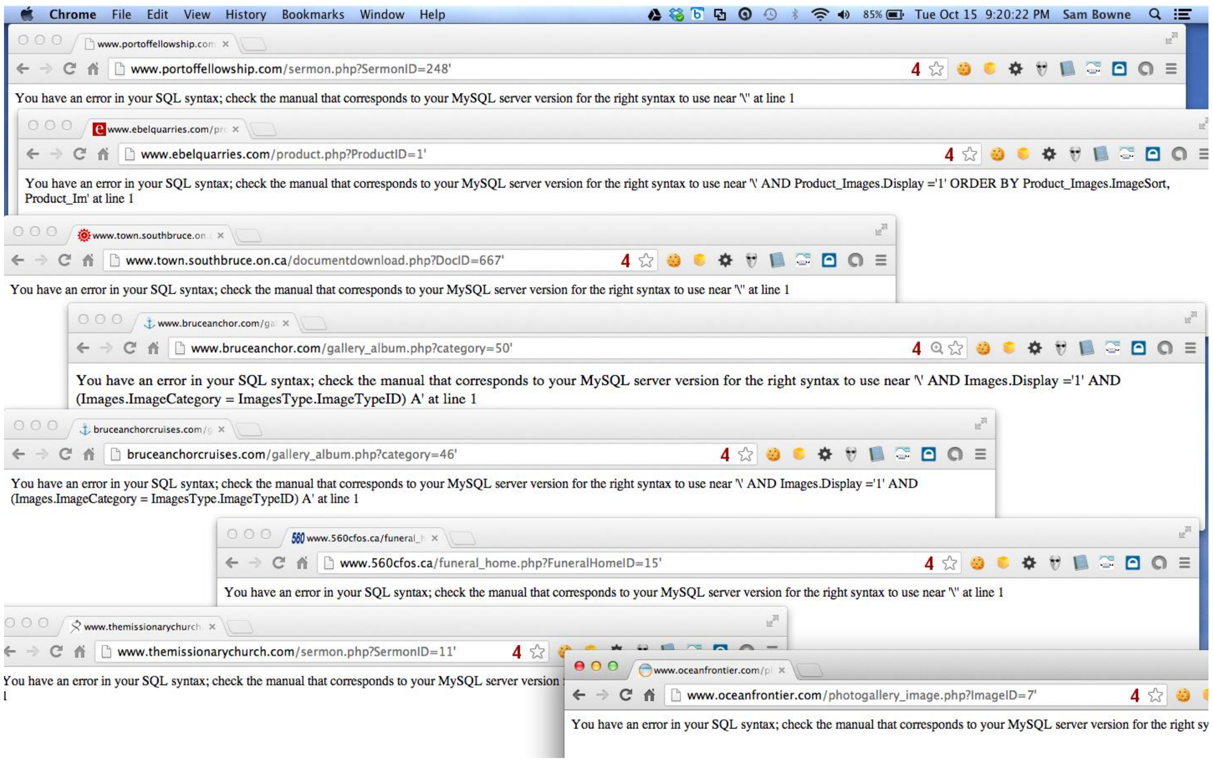
## 1. Introduction

Many organizations and companies have a problem to secure their web application and the database that connected with it, because of the hackers still can do many webs hacking through code injection attacks such as SQL injection attacks and XSS attacks, we build our framework to help organizations and companies to check if their web applications have any vulnerability that let hackers to exploit it to do their malicious attacks such as stealing the dataset of web application or make any sort of damage to the web application.
The basic idea of the paper is to build a framework that injects a code to the web site that we want to check wither is vulnerable or no, We need to specify what the meta-characters or metadata that we can use as an injection code to the website to check its vulnerability against SQL injection attacks and XSS attacks as it shows in table 1.

**Table 1.** SQL injection and XSS injection attacks Methods

| | SQL code injection | XSS code injection |
|---|---|---|
| 1. | '="or' | <script>alert('XSS attack')</script> |
| 2. | OR 1=1– | <IMG SRC="javascript:alert('XSS');"> |
| 3. | OR '0' = '0' | <script>alert(document.cookie)</script> |
| 4. | ') OR ('0' = '0' | %22sCrIpt%2Balert(%27XSS%27)%2B/sCrIpt %22 |
| 5. | UNION | <Img SRC=javascript:alert(String.fromCharCode(88,83,83))> |
| 6. | ' | <Img SRC=# Onmouseover="alert('xxs attack')"> |
| 7. | SHUTDOWN | <SCrIPt>Alert(String.fromCharCode(88,83,83))</SCrIPt> |

So to know whether the website is vulnerable to code injection attack or not, the framework injects a meta-code to the website, if the injection is success, it will show that there is some problem that happensin the website as it shown in figure 1, we can know if the website is vulnerable to code injection attack or not.


Chrome File Edit View History Bookmarks Window Help
85% Tue Oct 15 9:20:22 PM Sam Bowne

```
www.portoffellowship.com/sermon.php?SermonID=248'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1

www.ebelquarries.com/product.php?ProductID=1'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\' AND Product_Images.Display ='1' ORDER BY Product_Images.ImageSort, Product_Im' at line 1

www.town.southbruce.on.ca/documentdownload.php?DocID=667'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1

www.bruceanchor.com/gallery_album.php?category=50'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\' AND Images.Display ='1' AND (Images.ImageCategory = ImagesType.ImageTypeID) A' at line 1

bruceanchorcruises.com/gallery_album.php?category=46'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\' AND Images.Display ='1' AND (Images.ImageCategory = ImagesType.ImageTypeID) A' at line 1

www.560cfos.ca/funeral_home.php?FuneralHomeID=15'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1

www.themissionarychurch.com/sermon.php?SermonID=11'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
1

www.oceanfrontier.com/photogallery_image.php?ImageID=7'
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right sy
```

**Fig. 1**. Some websites that have a vulnerable against SQL injection attacks [11]

In figure 1 the attacker injected a meta-code (') to the web page that connected to the database of the web application to check whether is vulnerable to SQL injection attack, and the results show that there are errors in the SQL syntax and MySQL server, so it means that the websites are vulnerable to SQL injection attacks. We can know that the page of the web application is connected to a database by the URL link, if the URL link of the web application end up with a question mark sign (?) then an equal sign (=) then a number, it means that the web page is connected to the database of the web application.

## 2. Problem Statement

There are big challenges that the user is totally unaware about the CIA's into web browsers. Lots of companies and websites administrators are suffering from the fact that websites administrators do not know if the company website is vulnerable from CIA or not. Consequently, when a company wants to build a website, the web administrator does not know which protection program the company should use, as it depends on how much the company can afford to buy and use this protection system, or if it is easy enough for the web administrator to use the protection program that the company wants to use as its protection system. So the web administrator uses a protection system according to the cost of it or if the web administrator knows how to treat the protection system. But sometimes the web administrator does not know if the protection system that the company will use is really beneficial for the company to protect against CIA's [12]. So the problem is that a company or a system administrator are not sure if the protection system that they used to protect the website of the company can fully protect the website from CIA's. There are a lot of researchers talking about detection and protection from hacking attacks, and a lot of protection systems can provide some kind of protection against hacking attacks, however, there are no researchers looking into checking if the website is still vulnerable against CIA or not. This issue is important because users are facing this problem of needing an alert system in real life to take preventing measure against CIA's [12].

## 3. Literature Review

Fonsecaet al. [4] proposed The paper "Mapping Software Faults with Web Security Vulnerabilities", authors classified 655 SQL injection and XSS that they found them in 6 websites applications. Authors conclusion shows that a group of twelve faults in a generic software framework is responsible to all of the security troubles against code injection attacks such as SQL injection and Cross-site scripting (XSS). Authors detected MFC extended which is a part of a Missing Function Call fault, which is responsible of 76% for all the security analyzed problems of code injection attacks. By comparing the distribution of the Missing Function Call fault of the results with researches of software faults authors found considerable differences between them. The detailed analysis of the location or condition of each fault was observed and presented, so it supports the future works for definition the realistic of fault types which effect on detection the vulnerability in a web pages. In the future work, authors want to study the exploit code that an attackers used to Penetrate web pages, the results can help to create an attack injector to Penetrate web pages.

Stott et al. [7] proposed the paper "NFTAPE: a framework for assessing dependability in distributed systems with lightweight fault injectors", In this paper authors presented NFTAPE, that authors used it as an experiments to conduct fault injection automatically, they used it to help them solving these problems monitors, fault injectors, target specific, trigger, driver based, hardware based, debugger based, simulation based, and performance-fault. Authors used fault injection to accelerate the rate of errors happen in the computer system, for analyzing the dependability of the system, by presenting the weakness and evaluating the coverage of fault injection technique. Different tools of fault injection techniques have been presented. These injectors consist of simulated fault Injectors, physical fault injectors, and software implemented fault injectors (SWIFI). SWIFI is an inexpensive, easy technique to develop and runs in program. The architecture component helps to make it easy for using NFTAPE to many of operating system programs such as Windows, Solaris, Lynx, and Linux. Authors used NFTAPE to do experiments on fault injection technique. They presented 2 experiment examples of fault injection technique. The first experiment example authors inject bit errors into the physical layer of a LAN link by using a hardware fault injector. The second experiment example is framework of the real space imaging by using a fault injector of debugger-based. NFTAPE is flexible with an ability to execute simulated-based fault injector, SWIFI, or hardware-based fault injector. the contribution of the authors is the library of reusable components that built for NFTAPE.

Neveset al. [3] proposed the paper " Using Attack Injection to Discover New Vulnerabilities", in this paper authors present a tool called AJECT which is for discovering the security vulnerabilities on web servers/networks applications. AJECT simulates the behavior of a malicious attack by injecting many types of code injection attacks into the computer server target. AJECT produces malicious attacks automatically to perform the malicious attacks to the target computer server, while the application running, it monitors the framework to collect many sorts of information to determine whether the server executed incorrectly (vulnerability exists) or not. For evaluation, authors conducted several experiments with Internet Message Access Protocol (IMAP), which is a protocol used to access and preview electronic mail on a network of the computer server. The experiments show that AJECT utilized to locate distinct types of vulnerabilities such as format strings, information disclosure bugs, and buffer overflows). So AJECT can find many sorts of code injection vulnerabilities, such as the previously mentioned vulnerability.

Fonseca et al.[6] proposed the paper "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks", in this paper authors evaluate web pages vulnerability using commercial scanners, they injected a software faults in the web pages. Authors compare the efficiency of the different commercial scanners tools in the detection of the vulnerabilities. By analyzing the false positives and the detected vulnerabilities, authors evaluate and compare 3 of the commercial web page vulnerabilities scanners to know which one is better in detection of the code injection vulnerabilities in the web pages. the results show that the scanners get many different results but all of them produce a high false negative and false positives percentage values (false positives is in between 20% to 77%). In the future work authors want to apply and evaluate the same methodology to anther web pages that have not been used in this approach methodology, then compare many other approaches methodologies to get a bigger experiments results and to study which sort of programming code that leads to code injection vulnerabilities so it can support the prevention of code injection vulnerabilities.

Bau, et al. [1] proposed the paper "State of the Art: Automated Black-Box Web Application Vulnerability Testing", authors studied the vulnerabilities of black-box web application scanners to detect vulnerabilities of XSS and SQL injection, Cross-Site Request Forgery, Cross-Channel Scripting, Information Disclosure, and Malware. Authors goal is to identify the strengths and the limitations of different tools and to report test results of detect vulnerabilities using different tools. The eight commercial scanners tools that incorporated in authors study are (NeXpose 4.8.0, McAfee-SECURE, Rational AppScan 7.9, WVS 6.5, HailStorm-Pro 6.0, QA-Edition 7.0.0, QualysGuard-PCI, Web-Inspect 8.0), the study includes scanners tools of many of the companies in the web security. The prices of the scanners tools starts from (100's to 10,000's) of dollars, which gives a many options for users and customers to select which scanner tool is more appropriate for them. The scanners tools are poorly in detecting “stored” vulnerabilities, such as the stored XSS detection rate was 15%, scanners tools could not detect the vulnerabilities of the second method of SQL injection. 7 scanners tools detected the vulnerabilities of the first method of SQL injection (which is around 14%). Just one scanner tool has been able to get a high rate of detection the vulnerabilities of the first-order SQL injection (which is around 40%). Authors testbed has a server running Apache server 2.2.3, MySQL 5.0.45, PHP 5.1.6, and the PhpMyAdmin installed on a Linux 2.6.18 was running on the server with around 50 URLs, 3000 lines of code.

Bhojak, et al. [8] proposed the paper "Automated Web Application Vulnerability Detection With Penetration Testing", in this paper authors proposed a new approach to find a vulnerability in code injection attacks that include a black-box on some existing different web applications such as: Mutillidae, WebGoat, Swapp, dvwa. Authors performed tests to scan for all vulnerabilities using the commercial web vulnerability tools scanners, then they compare their open source scanner with the commercial web vulnerability tools scanners, and scan some web applications to get the results for different web vulnerability as presented in table 2.

**Table 2.** Results of comparison between commercial web vulnerability scanners and Authors Scanner

| | | Commercial Scanner 1 | Commercial Scanner 2 | Authors Scanner |
|---|---|---|---|---|
| Site tested | Vulnerabilities | Number of Vulnerability Detected | | |
| web applications Site 1 | Cross Site Scripting Attacks (XSS) | 5 | 7 | 8 |
| | SQL Injection Attacks | 8 | 7 | 7 |
| | Directory Listing | 11 | 9 | 12 |
| | Broken Authentication using SQL Injection | 1 | 1 | 1 |
| | Auto-complete enabled on input fields Enabled | 0 | 0 | 0 |
| web applications Site 2 | Cross Site Scripting Attacks(XSS) | 6 | 5 | 8 |
| | SQL Injection Attacks | 3 | 5 | 6 |
| | Directory Listing | 12 | 9 | 13 |
| | Broken Authentication using SQL Injection Attacks | 0 | 0 | 0 |
| | Auto-complete enabled on input fields Enabled | 1 | 0 | 2 |

According to table 2, it shows that the authors tool scanner in web applications Site 1 gives a better result in detection of vulnerabilities such as SQL Injection Attacks, Directory Listing comparing with Commercial Scanner 1,2, but the Commercial Scanner 1 gives a better result in detection Cross-Site Scripting Attacks (XSS) comparing with Authors Scanner. In web applications Site 2 the Authors Scanner gives a better results in detection SQL Injection attacks, Directory Listing, XSS attacks, Auto-complete enabled on input fields Enabled, while it gives the same result in detection Broken Authentication using SQL Injection Attacks for both of Commercial Scanner 1,2 and the Authors Scanner.

## 4. Methodology

The idea in our framework is different than other researchers works, our idea is to depend on building a framework that can do a code injection attacks to the web applications, so in this way, our framework can check wither the web application has a vulnerability against code injection attacks or not.

The methodology: In our program at first level, we enter the link of URL page that we want to check its vulnerability against SQL injection attacks and XSS attacks, then the program gives all pages link that connected with the URL page that we entered before. Soat this level, we will get all URL pages which connected with the URL page, because we need to get the pages that contain the databases of the URL page, so we can do a checking vulnerability in the next step level.

In the second level, we checking the vulnerability for all URL links by injecting a SQL injection code and XSS code attacks for every URL links, in this level we injected SQL code and XSS code such as:
injects[0] = "'="or'", injects[1] = "OR 1=1–", injects[2] = "'OR", injects[3] = "%22%2Balert(%27XSS%27)%2B%22", injects[4] = "OR '0'= '0'", injects[5] = "') OR ('0' = '0'", injects[6] = "'UNION'", injects[7] = "'WHERE'", injects[8] = "'".

In the last level, if the injected is successful, the program will not give an error about the code injection attacks otherwise, the code injection attack does not asuccess, so the website is not vulnerable.

We collect the dataset for checking the code vulnerability from different resources and the dataset is from real-world websites links.

The methodology of this paper is different than other methodologies that we presented before which is depend on checking the vulnerability of the web applications by doing a code injection attacks to the website not as other methodology which checking the commercial vulnerability scanners, or by make a protection framework against code injection attack by checking the source code of the same web page whether is vulnerable to code injection attack or not vulnerable.

## 5. Experimental Setup

In our experiment, we have collected the dataset from 10 open sources such as Antunes, et al., [13], The latest list of SQL injection vulnerable websites 2016 [14], Vulnerable Sites To (Legally) Practice Your Hacking Skills, [15], Download Biggest SQL Vulnerable Website List For 2017 [16], 150 SQL Vulnerable Websites 2017 List, August 2017 [17], List Of web sites Vulnerable For SQL Injection, August 2017 [18], 20 famous websites vulnerable to cross [19], Xss: xss vulnerable sites collection, August 2017 [20], 10000 fresh sqli vulnerable websites list, August 2017 [21].
We got the benign dataset which is around 100 line of code from the paper "Detecting SQL Injection Attacks Using SNORT IDS"[10].
We have 200 line of data code of the dataset we use for test our program framework, we built the program framework using a C# programming language, and we did the checking for each data line of code manually within the program framework.
Proposed Algorithm:
Step 1:
Get the URL web application that a tester wants to check the vulnerability of it and type it in the text box of the searchbutton.
Step 2:
Get all the URL's pages that connected with the URL web application that a tester entered in the text box of the searchbutton.
Step 3:
From step 2, a tester will get the pages that contain the database of the web application which a tester needs to arrive it to do the code injection attack.
Step 4:
Start checking the URL's pages by doing a code injection attack such as SQL injection attacks and XSS attacks.
Step 5:
A tester inject this code on step 4 such as: injects[0] = "'="or'", injects[1] = "OR 1=1–", injects[2] = "'OR", injects[3] = "%22sCrIpt%2Balert(%27XSS%27)%2B/sCrIpt%22", injects[4] = "OR '0'= '0'", injects[5] = "') OR ('0' = '0'", injects[6] = "'UNION'", injects[7] = "'WHERE'", injects[8] = "'", injects[9] = "''".
Step 6:
if the injected is successful, the program will not give an error about the code injection attacks. Otherwise, the code injection attack does not a success, so the website is not vulnerable.

### *5.1 Algorithm Evaluation*

In this subsection, we evaluate our algorithm from 10 different dataset sources, We have a dataset with 200 lines that we collected from different data sources, the results are:
TP: Attack detected where it was an attack
TN: not attack no alarm no detection
TNR= TN/(TN+FP)
Precision = Positive Predictive Value=TP/(TP+FP)=P
Recall = True Positive Rate = Sensitivity=TP/(TP+FN)=R
True Negative Rate is called Specificity.
False Positive Rate is 1 - Specificity.
Accuracy = (TP+TN)/(TP+TN+FP+FN)
TP=83
FP= 6
FN=0
Precision = positive predictive value=TP/(TP+FP)=83/(83+6)=0.9325
Recall = true positive rate = sensitivity=TP/(TP+FN)=83/(83+0)=1
TNR= TN/(TN+FP)=111/(111+6)=0.9487

Accuracy = (TP+TN)/(TP+TN+FP+FN)=(83+111)/(83+111+6+0)=0.9708.

## 6. Comparison with other Approaches

1. In the paper of "Training Security Assurance Teams using Vulnerability Injection", authors detecting the vulnerability of code injection by changing the code of web pages.
   The changes on the code of web pages do not affect on the running of the web page, the web will run without any errors in execution [5].
   One of these changes is executed, and it is depending on which code is surrounding the function of code, as it shown in the table 3.

**Table 3.** Changes on the code and the function of the code

| | Function of the Code | The Code | Example |
|---|---|---|---|
| 1. | If the function is used in an assignment as the only line of code and the variable is not inside a $_GET, $HTTP_GET_VARS, $_POST, $HTTP_POST_VARS PHP variable array | The whole line of code is removed. | Remove the line “$vuln_var = interval($vuln_var);”; |
| 2. | If the function is used in an assignment as the only line of code and the variable is inside a $_GET, $HTTP_GET_VARS, $_POST, $HTTP_POST_VARS PHP variable array | Only the function is removed from the code, leaving the argument intact. | Replace “$vuln_var = interval($_GET[‘vuln_var’]);” with “$vuln_var = $_GET[‘vuln_var’];”; |
| 3. | if the variables is inside, the $_GET, $HTTP_GET_VARS, $_POST, $HTTP_POST_VARS PHP variable array | leave the variables in the code, only the function is removed. | Replace “…“’str1’.intval($vuln_var). ’str2’”;” with …“’str1’.$vuln_var.’str2’”;”. |

   The authors find that the integer variables ("int" as in C programming language) are a main code injection vulnerability target. It has been collected from subtype A and responsible for 45% of all the vulnerabilities that have been founded by the authors. Authors also check if the "int" variables values are not have a non-integer values (Ex. display, assign). The variables that contain an integer values can be a target to the penetration of code injection attacks, and the penetration attack can be success by adding a meta-code to them, such as:
   A- Penetration of SQL Injection attacks , Ex. “ or 3=3”, “ or ’D’=’D’”.
   B- Penetration of XSS attacks “<scrIpT>Alert(’alert xss attack 1’)</scrIpT>”.
   C- Adding at the begin or at the end of the variables [”], [( ] ,[ )] , [ >], [<] ,[ ’].
   Authors used commercial vulnerabilities scanners for web pages to check if the commercial scanners can detect the vulnerabilities and asset the conclusion of results with the security penetration teams. Authors used the HP-WebInspect 7.7, and IBM Watchfire-AppScan 7.0 scanners and named them as S1 and S2, authors make it anonymous because of the commercial license does not allow the publication of evaluation of the commercial scanner.
   The Test Results:
   The precision rate of Commercial Vulnerability Scanners is 20%
   The precision rate of Security Teams after Basic Training Period is 30%
   The precision rate of Security Teams after Specific Training Period is 70%
   Code Inspection Results:
   The precision rate of Security Teams after Basic Training Period is 55.56%.
   The precision rate of Security Teams after Vulnerabilities Specific Training Period is 100%.
   The limitation is that authors did not build a framework for detection the vulnerability of code injection attacks in the WebPages, they test a Commercial Vulnerability Scanners, and training a Security Teams for detection vulnerability of code injection attacks in the WebPages.
2. In the paper of "Automatic Detection and Correction of Web Application Vulnerabilities using Data Mining to Predict False Positives", Web applications have been the biggest source of problems in the security point of view as some of the reports that indicate the increase of web applications attacks in 2012 is around 33%. The main reason for the increasing of web pages attacks is that lot of the users do not have enough knowledge about how to secure the web pages. The authors present a methodology to protect web pages, the methodology is depending on analyze the source code of the web pages then detecting the code

injection vulnerabilities and modifying the source code in the way of correcting the code injection vulnerabilities [9].
The contribution for the web applications security is by fixing the vulnerabilities, and the programmers can learn from their mistakes by practicing and finding the code injection vulnerabilities.
Authors use Static analysis mechanism with a hybrid method to detect code injection vulnerability. That gives a better effective way to detect the code injection vulnerabilities in source code, but it could be reporting many false positives.
The designing and implementing of WAP (Web Application Protection) is consisting of analyzing the PHP programs. Attacker enter the web page by using an entry points (for example: $ GET) then the attacker Penetrate the vulnerabilities through a database query language such as (MySQL query). Many of the attackers using a benign inputs with meta-characters or meta-data such as ( ', OR). The web pages is protecting by adding sanitization function between the entry points ($ GET) and the database query language (MySQL query).
Table 4 shown the Sanitization function that have been used to remove the SQL injection vulnerability, which is provided by PHP programming language.

**Table 4.** Sanitization functions used to fix PHP code for SQLI vulnerabilities.

| Vulnerability | Entry points | Sensitive sinks | Sanitization functions |
|---|---|---|---|
| SQL Injection Vulnerability | $Get<br>$Post<br>$Cookie<br>SQLI $ REQUEST<br>HTTP-Get-VARS<br>HTTP-Post-VARS<br>HTTP-Cookie-VARS<br>HTTP-Request-VARS | mysql-Query<br>mysql-unbuffered-Query<br>mysql-db-Query | mysql-real-escape-String |
| | | mysqli-Query<br>mysqli-real-Query<br>mysqli-master-Query<br>mysqli-multi-Query | mysql-real-escape-String |
| | | mysqli-stmt-execute<br>mysqli-execute | mysqli-stmt-bind-Param |

To check the result if it is false positive or true positive, authors involved getting the source code of the web page and doing code injection attacks for all vulnerabilities. By correcting and validating the data in the webpage source code, authors can stop the bad effect of meta-characters.
The conclusion of the research paper that contain of the 2 framework tools, nine open source programs and the PHP code of NIST.
The reference of SAMATE Dataset is from [http://samate.nist.gov/SRD/].
WAP's detector detected sixty eight vulnerabilities (twenty two of SQL injection attacks and forty six of XSS attacks), and twenty one of false positives. Pixy detected seventy three vulnerabilities (twenty of SQL injection attacks and fifty three of XSS attacks), and forty one of false positives and five of false negatives, WAP-TA detect five vulnerabilities while the other detectors did not detect them).
The Pixy's accuracy is around 44%, WAP-TA's accuracy is around 69%, and WAP's accuracy is around 92%. The limitation is the high false positive rate which is 20%.

3. In the paper of "Vulnerability & Attack Injection for Web Applications", Authors proposed a code injection approach to detect the vulnerabilities of web pages, they used many programs to applied their idea, that gives a precision methodology to check whether the web application is secure or not [2]. Authors used a case studies depend on the real world to validate the ability of the framework to penetrate web pages, so it validate an Intrusion Detection System and 2 web page code injection vulnerability Scanners for SQL injection attacks.
The Dependency Builder is the first component of the code injection program. Dependency Builder is seeking for the files within the input file, the Dependency Builder is targeting PHP file for the injection process of code injection attack.
In PHP program, programmers can include a general file in to another file, (it can be done using one of this statements: require(), include(), require_once(), include_once()) for reutilization Purposes.
The execution of the main file and the included files is processing by the PHP tool and to search on vulnerabilities, meta-code of code injection attack should be injected.
The second component is the Variable Analyzer. it analyzes all the variables of the SQL queries. The Variable Analyzer collects all PHP variables from the PHP code, then the Variable Analyzer seeks on PHP variables in the SQL query.
Table 5 presents queries that injected sent to the database and execution of SQL Injection. It is consist of the basic of SQL Injection attack strings.

**Table 5.** Basic attack payload string examples

| Attack Payload | Strings Expected result of the attack |
|---|---|
| ‘ | The query result is an error |
| Or 2=2 | The query result is the override of the query restrictions |
| ‘ Or ‘b’=’b | The query result is the override of the query restrictions |
| +connection-id<br>-connection-id | The query result is 0 |
| +2-2 | The query result is 0 |
| +67-ASCII('A') | The query result is 0 |
| +51-ASCII(1) | The query result is 0 |

To evaluate the code injection attack tool for Experimental results authors create 3 groups of experiments. In the first group, they injected meta-code into 3 web pages to validate the performance of the attack. In the second group, they tested an IDS for databases so they can validate the efficiency of the intrusion detection system in the detection of code injection attacks that have been injected by the injector.
In the third group, Authors assess 2 web page vulnerability scanners depending on the ability to detect the code injection vulnerabilities.
In the experiments, authors used LAMP (Linux, Apache, Mysql and PHP) approach. The web server Apache works within the Linux server. Authors used three web applications, the first web application is the TikiWiki that used to build wiki which is a website that let users to contribute to it using adding method or modifying method to its contents).
The second web application is the phpBB. which is a LAMP application, and it is used as an Open Source forum solutions.
The third web application is MyReferences. MyReferences consists of thirteen PHP files and it can be used for management of publications such as the conference, the title, the year of publication.
The result shows that the approach is effective in code injection attack tasks, and the intrusion detection system could detect most of the penetration attacks but the approach gets a high rate of false positives and the code injection vulnerability scanners do have some difficulties in detecting most of the vulnerabilities of code injected attacks.
Authors work is close to our work but the limitation is that they did not do the injection or detection tools in a programming framework, they work on commercial scanners tool and they do a code injection attacks manually.

## 7. Conclusion

This paper presents a framework method to identify the susceptible websites against code injection attacks. It gives a precision performance compared with other works in the same field as this paper showed before. In this paper, the algorithm framework detects various types of CIA's such as SQL injection attacks, Cross Site Scripting attacks (XSS).
The dataset that we used in this paper has been collected from 10 different resources, we collected 200 lines of code data in the dataset that we use for testing our frame work and applied our framework to them to get the performance of our framework.
The accuracy result of this paper is 0.9708%, with a precious performance of precision rate = 0.9325 and recall rate =1, while the other papers one of them gives an accuracy result was 92.1%.
Our idea is different than other paper work such as [1, 5, 9], which depending on injecting a meta-code by the framework on the web application that we want to check its vulnerability against code injection attacks, such as inject SQL injection meta-characters and XSS meta-characters to check whether the website is vulnerable to code injection attack or not, while the other papers depend on checking the vulnerability of web application by using a commercial applications that have been downloaded from the internet, or by checking the source code of the web application and do some modification on the source code of the web application to prevent the vulnerability against code injection attacks.

## References

1. Bau, J., Bursztein, E., Gupta, D., Mitchell, J. (2010). State of the Art: Automated Black-Box Web Application Vulnerability Testing, SP '10 Proceedings of the 2010 IEEE Symposium on Security and Privacy,(pp. 332--345). IEEE.

2. Fonseca, J., Vieira, M., Madeira, H. (2009)Vulnerability & Attack Injection for Web Applications, Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE.

3. Neves, N., Antunes, J., Correia, M., Verissimo, P., Neves R. (2006). Using Attack Injection to Discover New Vulnerabilities, IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE.

4. Fonseca, J., Vieira, M. (2008) Mapping Software Faults with Web Security Vulnerabilities", IEEE/IFIP Int. Conference on Dependable Systems and Networks, IEEE.

5. Fonseca, J., Vieira, M., Madeira, H. (2008). Training Security Assurance Teams using Vulnerability Injection, IEEE Pacific Rim Dependable Computing conference, IEEE.

6. Fonseca, J., Vieira, M., Madeira, H. (2008). Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks. Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, (pp.365—372). IEEE.

7. Stott, D.T., Floering, B., Burke, D., Kalbarczpk, Z., Iyer, R.K. (2000). NFTAPE: a framework for assessing dependability in distributed systems with lightweight fault injectors, Computer Performance and Dependability Symp., IEEE.

8. Bhojak, P., Shah, V., Patel, K., Gol, D. (2017).Automated Web Application Vulnerability Detection With Penetration Testing. ICRISET2017 International Conference on Research and Innovations in Science, Engineering &Technology, Vol. 2, (pp.177—187). Kalpa.

9. Medeiros, I., Neves, N., Correia, M. (2014). Automatic Detection and Correction of Web Application Vulnerabilities using Data Mining to Predict False Positives, WWW '14 Proceedings of the 23rd international conference on world wide web, (pp. 63—74). IEEE.

10. Alnabulsi, H., Islam, R., Mamun, Q. (2014). Detecting SQL Injection Attacks Using SNORT IDS. Asia-Pacific World Congress on Computer Science and Engineering Conference, (pp. 1--7). IEEE.

11. Websmart, Inc. and 100,000 Vulnerable Websites, https://samsclass.info/125/ethics/smart-websites.htm, September, 2017.

12. Hussein Anabulsi, A Detection Framework against Code Injection Attacks, Charles Sturt University.

13. Antunes, N., Vieira, M., Madeira, H., "Web Services Vulnerabilities", 2008, http://eden.dei.uc.pt/~mvieir

14. The latest list of SQL injection vulnerable websites 2016, August 2017, http://www.sunnytech7.com/pages/latest-2016-list-of-SQL-vulnerable-websites.html

15. Vulnerable Sites To (Legally) Practice Your Hacking Skills, August 2017, https:// www.checkmarx.com/2015/04/16/15-vulnerable-sites-to-legally-practice-your-hacking-skills/

16. Download Biggest SQL Vulnerable Website List For 2017, August 2017, http://www.arkdots.com/sql-vulnerable-website-list/

17. 150 SQL Vulnerable Websites 2017 List, August 2017, https:// www.scribd.com /document/ 325850327/ 150-SQL-Vulnerable-Websites-2017-List

18. List Of web sites Vulnerable For SQL Injection, August 2017, http://listsql.blogspot.com.au/

19. 20 famous websites vulnerable to cross, August 2017, http://thehackernews.com/2011/09/20-famous-websites-vulnerable-to-cross.html

20. Xss: xss vulnerable sites collection, August 2017, http://cyb3rninjas.blogspot.com.au/2015/03/xss-xss-vulnerable-sites-collection.html

21. 10000 fresh sqli vulnerable websites list, August 2017, https://pastebin.com/ATJE7VdZ

.