Analyzing & Defining Web Application Vulnerabilities With Dynamic Analysis And Web Mining

[1]Deepak B. Jadhav, [2]Sachin K. Sanap, [3]Ramesh C. Ghuge, [4]Deore Somnath

[1,2,3,4]UG Student, Department Of Computer Engineering, Late. G.N. Sapkal Collage of Engineering, Nashik, Maharashtra, India.

[1]*deepakjadhav@gmail.com*, [2]*sanapsachin72@gmail.com*, [3]*rameshghuge300@gmail.com*, [4]*somnath.sd23@gmail.com*

Abstract **Security is a critical part of your Web applications. Web applications by definition allow users access to a central resource the Web server and through it, to others such as database servers. By understanding and implementing proper security measures, you guard your own resources as well as provide a secure environment in which your users are comfortable working with your application. Web application security is a branch of Information Security that deals specifically with security of websites, web applications and web services. Web Security blocks web threats to reduce malware infections, decrease help desk incidents and free up valuable IT resources. It has more than 100 security and filtering categories, hundreds of web application and protocol controls, and 60-plus reports with customization and role-based access. You can easily upgrade to Web Security Gateway when desired to get social media controls, SSL inspection, data loss prevention (DLP) and inline, real-time security from Websense ACE (Advanced Classification Engine).**

Keywords: Data mining, Input validation vulnerabilities, software security, Web vulnerability security, vulnerability detection, advanced confusion matrix.

I. INTRODUCTION

A. WEB APPLICATION SECURITY

Although a large research effort on web application security has been going on for more than a decade, the security of web applications continues to be a challenging problem [1]. An important part of that problem derives from vulnerable source code, often written in unsafe languages. Source code static analysis tools are a solution to find vulnerabilities, but they tend to generate false positives, and require considerable effort for programmers to manually fix the code [2]. We explore the use of a combination of methods to discover vulnerabilities in source code with fewer false positives [6][7][5][3]. We combine taint analysis, which finds candidate vulnerabilities, with data mining, to predict the existence of false positives. Given this enhanced form of detection, we propose doing automatic code correction by inserting fixes in the source code [2][3][8]. web applications have become increasingly popular for delivering security critical services Because web applications are exposed to various threats and attacks, numerous tools, including commercial tools and open source software, have been developed for detecting web application vulnerabilities, called web vulnerability scanner [3].

B. WEB VULNERABILITY

Vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application. Stakeholders include the application owner, application users, and other entities

that rely on the application. The term "vulnerability" is often used very loosely. The Top Ten Vulnerabilities [1] that are updated by Open Web Application Security Projects are: Sql Injection: Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as trick the interpreter into executing unintended commands or accessing data without proper authorization.

II. LITERATURE SURVEY

A. Detection of Vulnerabilities by Security Scanner

Security scanners identify defects and weaknesses by a collection of signatures of known vulnerabilities. These signatures are updated regularly as new vulnerabilities are

discovered. In the search for vulnerabilities like XSS and SQL injection, the scanners execute lots of pattern variations adapted to the specific test in order to discover the vulnerability. There are two main approaches to testing web applications for vulnerabilities: “white box” and “black box”. The “white box” approach consists of the analysis of the source code of the web application. Static code analysis is a type of white-box analysis. This can be done manually or by using code analysis tools like FORTIFY, Pixy, Code Secure, etc. These static analyzer tools analyze source code to detect vulnerabilities, such as SQL injection and cross-site scripting. The black-box vulnerability scanner without knowing the internal design of the web application uses fuzzy techniques over the web HTTP requests, simulates numerous scenarios such as hackers’ intentional attacks or general users’ inadvertent attacks, and provides an automatic way to search for vulnerabilities, avoiding the repetitive and tedious task of doing hundreds or even thousands of tests by hand for each vulnerability type. There are many commercial web vulnerability scanners for black-box testing such as Acunetix Web Vulnerability Scanner, HP WebInspect [8], IBM AppScan [9]. The testing results of vulnerabilities for web applications are quite different from scanner to scanner. According to the survey presented in, black-box testing is the second most used technique to evaluate the effectiveness of security. In our experiments, we use our proposed approach to evaluate four popular “black box” commercial scanners, AppScan, WebInspect, Paros.

B. Web Vulnerability Scanner Evaluation Criteria

Vulnerability scanners are considered as a solution for detecting vulnerabilities and security threats in web applications. Among the studies focusing on tools evaluations, the Web Application Security Consortium proposed "Web Application Security Scanner Evaluation Criteria (WASSEC)"[25] project to provide a set of detailed evaluation criteria and a framework for conducting a formal scanner evaluation. The goal of the WASSEC is that for the tools given to users they need to conduct a solid evaluation and make their own informed decision on which scanner(s) best meet their needs. P.E. Black et al. proposed guidelines for describe the functional specifications of Source Code Security Analysis Tool and Web Application Security Scanner [6]. Through the development of tool functional specifications, test suites and tool metrics, the NIST Software Assurance Metrics and Tool Evaluation (SAMATE) project aims to better quantify the state of the art for different classes of software security assurance tools. The documents constitute a specification for a particular type of software assurance tool, which is referred to here as a web application security scanner. By examining the steps in scanning processes, we can reasonably assume that costs of vulnerability scanner include construction cost, operation cost, and analysis cost, with operation and analysis ones being the main parts in cost evaluation. These processes are generally labor-intensive and often involve substantial human resources, including developers, domain experts, and security experts. There are several studies focusing on reducing cost in vulnerabilities detection, such as [22][12]. However, the issue of redundant alerts have not been considered in the previous research. In general, when a certain defect is found repeatedly, the developers would spend double effort to solve it. In this paper, we proposed a cost-effective evaluation approach to evaluate vulnerability scanner by considering issue of redundant alerts [7].

C. Confusion Matrix

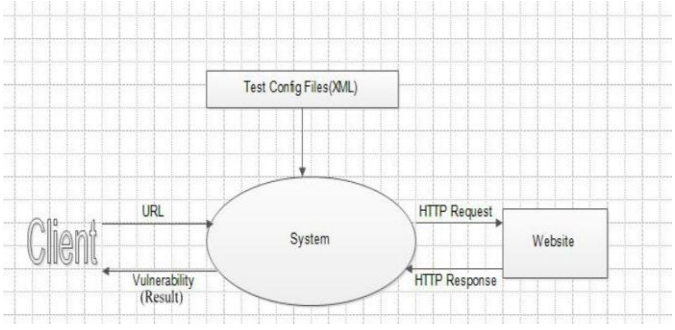
Previous research provides relative benchmarking measures for improvement and tuning of prediction errors, false positive and true positive, despite the definition of the benchmark measures is different. To compare the effectiveness of tools that implement different vulnerability detection approaches, Van Rijsbergen proposed the Fmeasured method that can be applied to characterize vulnerability detection tools. In fact, it represents the harmonic mean of two very popular measures (precision and recall) [8].

III. SYSTEM ARCHITECTURE

we implement the Web Vulnerability Scanner Testbed (abbrev. W-VST) to evaluate the performance of web vulnerability scanners based on our proposed approach. The W-VST is a web-based application and developed in JSP and MySQL. The basic idea of W-VST is to apply our cost-effective evaluation approach and support the security engineers to evaluate vulnerability scanners. In W-VST, we propose three vulnerable applications, WebGoat, WordPress, and, WackoPicko, as testing targets based on OWASPBWA project. OWASPBWA is a collection of vulnerable web applications [1][5][10].

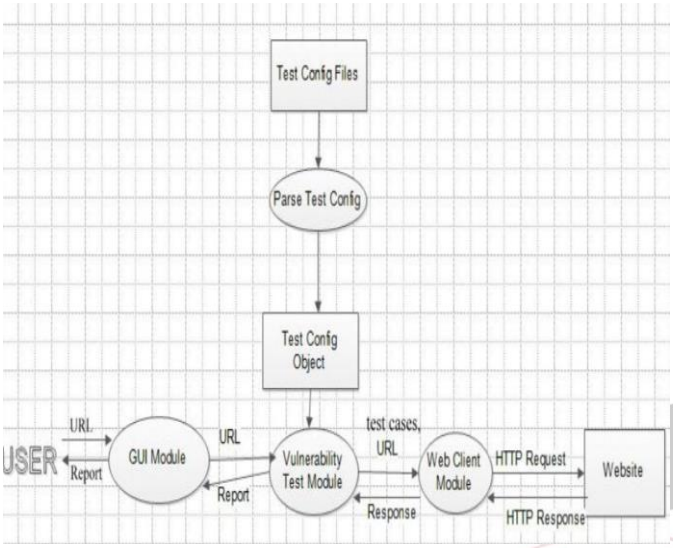
A. DATA FLOW DIAGRAM (DFD):

a) Level 0 : Context



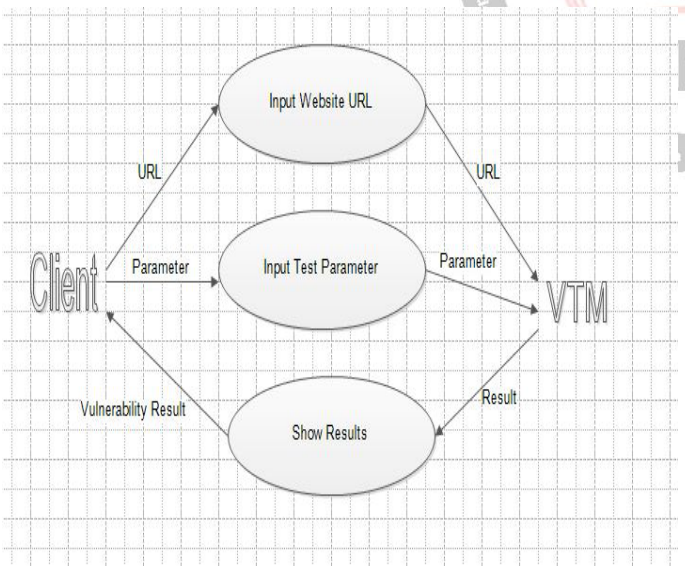
Level 0 shows the over view of the software. The client will provide url of the website to be tested and what kind of test to be performed on it. The system will build the crafted request with the help of test criteria's. The build crafted request is send to the server and scanner analyzes the response return by the web server and report will be send to client.

b) Level 1: System Architecture



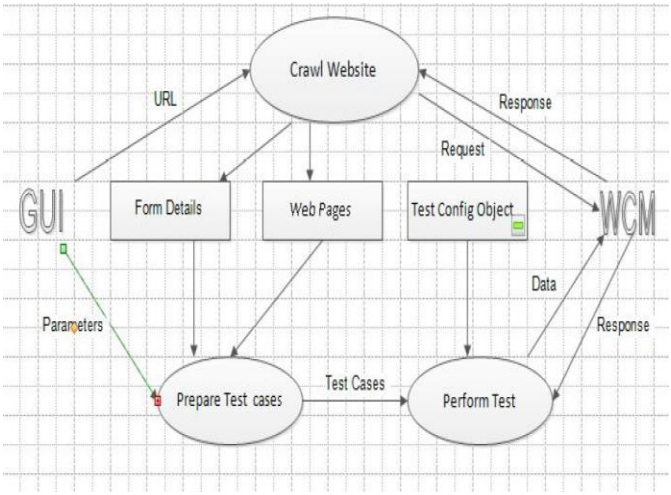
Level 1 is basically the architecture of the system i.e software .It has three module i.e. GUI Module Vulnerability Test Module, Website Client Module.

c) Level 2: GUI MODULE



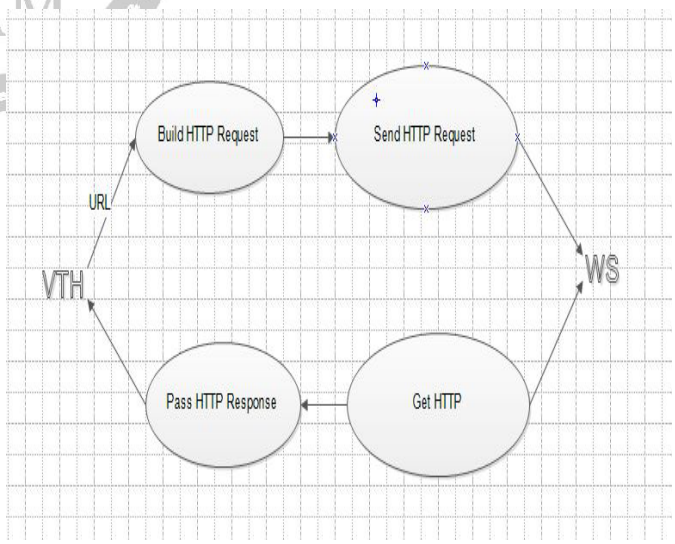
Graphical User Interface Module is basically a user interface of the software where user will input the url of the website and test to be performed on it. After the test is completed the result is shown here[6][8][15].

d) VULNERABILITY TEST MODULE



Vulnerability Test Module is a module in which crawling of the website, preparing test cases and analyzing the response that is sent by the web server is done. For eg. To check whether SQL injection attacks are possible, the vulnerability scanners send modified requests and analyze the responses returned by the server [3][4][8][9]. A server may respond with a rejection page or with an execution page. A rejection page corresponds to the detection of syntactically incorrect or invalid inputs [12][13][14]. An execution page is returned by the server as a consequence of a successful execution of the request. This page legitimate use of the web site, but may also result from a successful exploitation of an injection attack[12].

e) WEB-CLIENT MODULE



In this module, the scanner will build the crafted HttpRequest with the help of test cases and it is sent to web server and in return web server sends response to Web-Client Module and it than parses the response and sends the parse response to the Vulnerability Test Module where the response is analysed [8].

IV. ALGORITHM

1. Start
2. Enter the URL.
3. Select the type of vulnerability you want to scan for:
Default: checks for sql injection and cross site scripting both.
Sql Injection: checks for sql injection.
XSS: checks for cross site scripting.
4. Start scanning.
5. Check for the given conditions according to the selected Vulnerability scan.
6. After observing the conditions of the selected vulnerability prepare a report.

IV. RESULT ANALYSIS

A. URL Input:

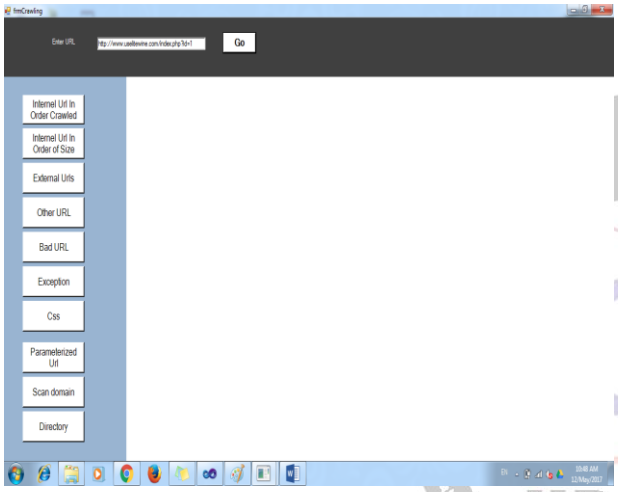


Figure. 1 URL Input

This is the home page of application. In this as seen there are different modules for checking vulnerabilities and one URL input section. Then give the input website URL to be tested for the different vulnerabilities types.

B. Internal URL in order of size:

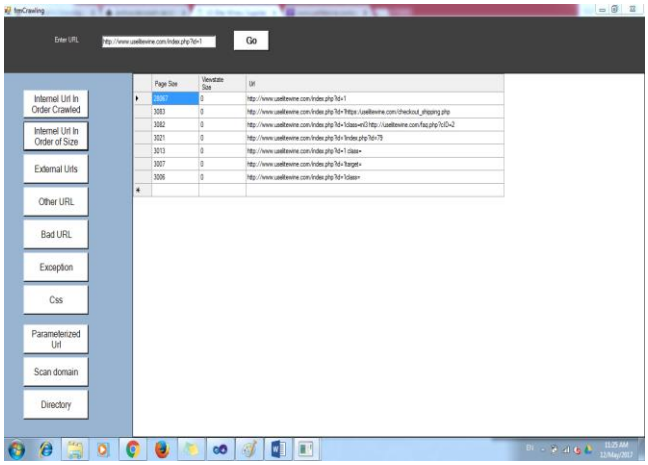


Figure. 2 Internal URL in order of size

The module Internal URL in order of size gives the details of the internal URL of website with the size. This information may be vulnerable so one must provide the security to this.

C. Parameterized URL

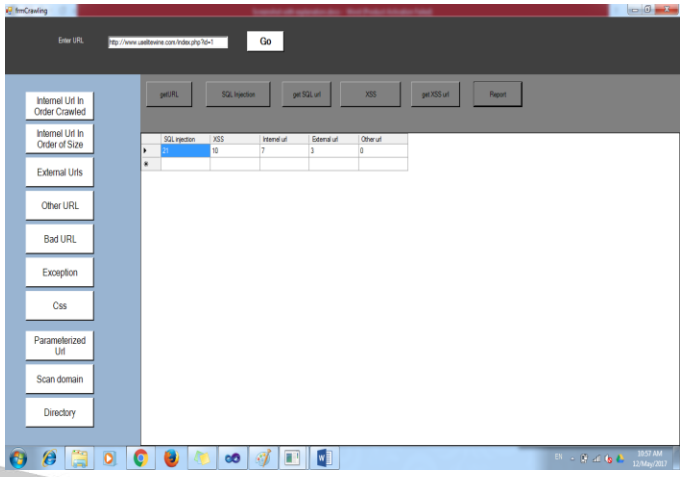


Fig. 3 Parameterized URL

In the parameterized URL module there are five records of vulnerabilities we can get and last one is the report section where we can have the all five records with number of vulnerabilities possible. So, this module is useful to check if there any SQL injections and XSS are possible or not.

D. Scan Domain:

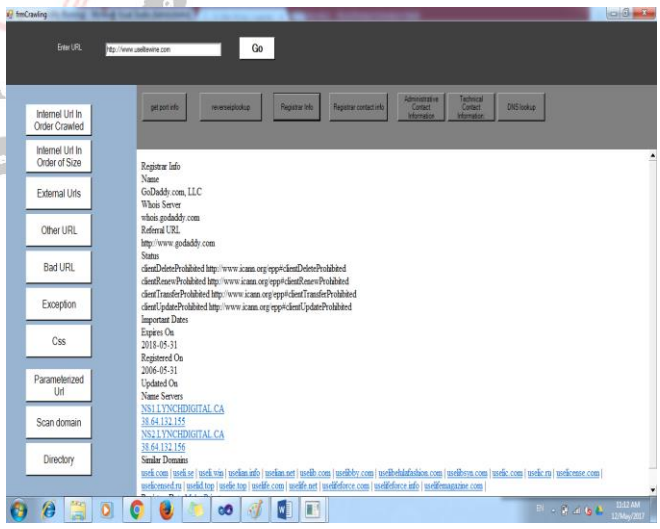


Figure 4 Scan Domain

The scan domain module have seven records of vulnerabilities. In get port info the port information can get if no security is provided. And Registrar info block gives all the information regarding domain, hostname, storage server, registration date, expiry date, updated on. Etc.

V. CONCLUSION

The relationship between detected and actual vulnerabilities is more complex than previously known, and it often requires many domain experts to participate in identifying the redundant detections. In this paper, we proposed a cost-effective approach to evaluating the performance of Web vulnerability scanner which we applied to our evaluation testbed. All detected and actual vulnerabilities can be handled in our proposed approach. Two experiments are performed with our testbed. The experimental results show that our approach can verify the performance in vulnerability scanner evaluation. In the near future, we will continue to increase features of scanner evaluation for the purpose of better simulating hacker's behaviors. Meanwhile we will also develop new evaluation techniques to shorten the time needed for scanner evaluation. In the future studies we will also explore the effect of integrating multiple detection results by several tools, which may further increase the accuracy.

ACKNOWLEDGMENTS

It gives us great pleasure in presenting the Result Paper on 'ANALYSING AND DEFINING WEB APPLICATION VULNERABILITIES WITH DYNAMIC ANALYSIS AND WEB MINING.'

I would like to take this opportunity to thank my internal guide Prof. R. D. More for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their Valuable suggestion were very helpful

REFERENCES

[1] "Acunetix Ltd, Web Vulnerability Scanner", http://www.acunetix.com/vulnerability-scanner/[Last accessed 31 May, 2014]

[2] C. J. van Rijsbergen, Information Retrieval, Butterworth, 1979.

[3] Confusion Matrix, http://en.wikipedia.org/ wiki/Confusion_matrix. [Last accessed 31 May, 2014]

[4] D. Stuttard and M. Pinto, The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. Wiley, 2007.

[5] F. Yu, M. Alkhalaf, and T. Bultan, "Stranger: An automata-based string analysis tool for php," in TACAS, 2010, pp. 154–157.

[6] Gray, J. (Ed.), "The Benchmark Handbook", Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.

[7] "HP FORTIFY", http://www8.hp.com /us/en/softwaresolutions/software.html?compURI=1337262#. UYKTGaLAdmx

[8] "HP WebInspect", https://download.hpsmartupdate.com /webinspect/.

[9] "IBM AppScan", http://www- 03.ibm.com/ software/products/us/en/appscan,

[10] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for sql injection and xss attacks," in PRDC, 2007, pp. 365–372.

[11] L. Gordon, M. Loeb, W. Lucyshyn, R. Richardson, "Computer crime and security survey", Computer Security Institute, 2006.

[12] L. Deepak Subramanian, Ha Thanh and L. Peter, Kok Keong, "Fuzzy heuristic design for diagnosis of web-based vulnerabilities," in Fourth International Conference on Internet Monitoring and Protection, 2009, pp. 103–108.

[13] Larry Suto, Analyzing the Accuracy and Time Costs of Web Application Security Scanners, San Francisco, 2010, http://ha.ckers.org/files/Accuracy-and-Time-Costs-of-Web-App- Scanners.pdf, [Last accessed 31 May, 2014]

[14] N. Antunes and M. Vieira, "Benchmarking vulnerability detection tools for web services," in ICWS, 2010, pp. 203–210.

[15] N. Jovanovic, C. Krugel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities (short paper)," in IEEE Symposium on Security and Privacy, 2006, pp. 258–263.

