Web Unique Method (WUM): An Open Source Blackbox Scanner for Detecting Web Vulnerabilities

Muhammad Noman khalid[1], Muhammad Iqbal[1], Muhammad Talha Alam[1], Vishal Jain[2], Hira Mirza[3] and Kamran Rasheed[1]

[1]Department of Computer Science, Bahria University, Karachi, Pakistan
[2]BVICAM, New Delhi, India
[3]Department of Computer Science, PAF KIET, Karachi, Pakistan

***Abstract*—The internet has provided a vast range of benefits to society, and empowering people in a variety of ways. Due to incredible growth of Internet usage in past 2 decades, everyday a number of new Web applications are also becoming a part of World Wide Web. The distributed and open nature of internet attracts hackers to interrupt the smooth services of web applications. Some of the famous web application vulnerabilities are SQL Injection, Cross Site Scripting (XSS) and Cross Site request Forgery (CSRF). We believe that in order to encounter these vulnerabilities; the web application vulnerabilities scanner should have strong detection and prevention rules to ease the problem. At present, a number of web application vulnerabilities scanners have been proposed by research community, such as ZED Attack Proxy (ZAP) by AWASP, Wapiti by sourceforge.net and w3af by w3af.org. However, these scanners cannot challenge all web vulnerabilities. This research proposed and develop a vulnerability scanning tool WUM (web unique method) to detection and prevention of all the major instance vulnerabilities and demonstrates how to detect unauthorized access by finding vulnerabilities. With the efficient use of this tool, the developers are able to find potentially vulnerable web application. WUM generated a high level of accuracy and compatibility, which is elaborated underneath. The result of the experiment shows proposed vulnerability scanner tool WUM which gives less false positive and detect more vulnerabilities in comparison of well-known black box scanners.**

***Keywords*—Automated vulnerability detection; black-box scanners; web vulnerabilities crawling; security scanner**

I. Introduction

Web applications are the best way of providing standard facilities through Internet. The collaboration of diverse technologies that are used in many generalization layers, are the foundation cause of vulnerabilities in web applications [1]. In fact, the number of reported website vulnerabilities is increasing abruptly. This could be minimized by providing a firm knowledge of web developers, or through security-aware web application development frames. This can be imposed by splitting the structure and input/output data of content [2].

Fig. 1 provides the unique and extensively used web application architecture of three-tier with help of each tier methodologies and modules of software. Advance features that intensify the intricacy of web application are given by technologies and architecture of web application [1]. With the popularity of forums, web services and blogging, attackers started taking interest in web applications [3]. A user getting a bug, loophole and weakness existing in the web application that can be exploited by an illegal user is known as vulnerability. Usually these vulnerabilities do several incursions for getting full command over web application. Globally renowned organizations have a serious issue regarding vulnerability system [4].

According to OWASP [5], the most Dangers web vulnerabilities are include XSS, CSRF and SQLi, among others. The information of these vulnerabilities exploits represent a significant threat for website and demand to secure these vulnerabilities with security counter measures. In order to overcome security breaches with successful attacks against web applications different penetration tester used variety of techniques around the globe. Many techniques assist to identify the vulnerabilities existing in the web application in order to prevent and minimize potential of web damages. However, testing the web application requires sufficient and experienced tester. An Additional burden is the fact that the testing process itself is a manual and prolonged process with essential requirement for precision [6].

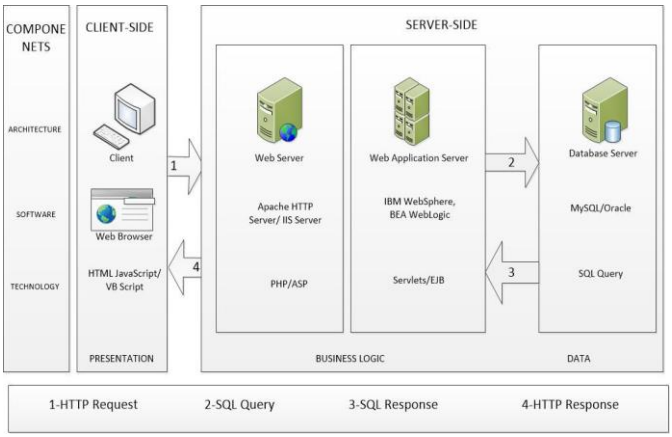

Fig. 1. Overview of Web architecture.

In order to support testers there is another methodology for analyzing the web vulnerabilities in web applications. This technique is to check the output of the application by providing some input to that specific output. This analysis method is said to be a black-box testing. There are a lot of automated and manual testing tools for XSS detection, SQLi also detecting other vulnerabilities scanners [6], [9]-[15] in order to make web security easier for web developers/admin. This research

presents an open source web vulnerability scanner that use black box technique to carry out crawling and scanning for websites, to effectively detect the presence of exploitable web vulnerabilities. This tool is independent of a database of known vulnerabilities; instead distinctive, underlying properties of application level vulnerabilities are exploited to effectively detect affected programs. It additively attempts to automatically generate proof of concept exploit in certain cases which serves to bring increment in confidence of the correctness of our scan results.

WUM architecture which is flexible incorporates multithreaded crawling, attack and analysis components. Employing the assistance of a graphical user interface the user can effectively configure single or combined crawling attack runs. Proposed prototype implementation, we currently provide various attack components, reflected and Stored XSS, SQLI, CSRF, LFI/ RFI, CJ, SSL and UR. In addition, an application programming interface is provided to enable the developers to implement their own modules in order to effectively launch their desired attacks. The main objective of this paper to gauge performance of black box scanner to detect web vulnerabilities and to detect rate can be improved. This research paper proposed a tool to increase the efficiency of black-box web vulnerability scanners by growing their ability to recognize the internal state of web application. The usability of that model is to down the application in a state aware mode, negotiating more of the web application. As a result, it finds out more vulnerabilities and provide flexibility for improvements, which helps to carryout penetration testing in more effective and efficient way.

The Remainder of this research paper is organized as follows. Section 2 describes types of web vulnerabilities and related work. Section 3 describes the methodology of WUM. Section 4, this section concludes the paper and present comparison between WUM and existing commercial open source vulnerability scanner.

II. Background and Litrature Review

A lot of work is being carried out by renowned organization, computer security enterprises, threat intelligence software companies, and independent security researchers for cyber vulnerabilities. Methods of Detection and prevention of web vulnerabilities have been studied widely. Machine learning, dynamic, static and combined are most preferable techniques All web based penetration testing scanners can easily be separated in three categories; i.e. academic, open source Scanner and commercial [7], [8]. Individuals having same interest of research take assistance from academic scanner to introduce their own scanner like SQIVS [9], Increase the MySQLinj factor [10], secubat [11] State aware scanner [12], Amnisia [13] and wave [14], etc. Many academic scanners are not in the reach of a public, language dependent and are under development. Thus, those techniques that are used in development cycle of defined scanners are publically present to ponder the light to every individual or academic researcher; those researchers who really need to upgrade the previous Scanners or launched new methodologies with advanced key factors.

Academic scanner public has access open source scanner like nikito, zap, wapit, vega and wa3p [15] for free over the copyright tags and policy. Thus, the framework, algorithm or development cycles are not accessible for public Use. Just like individuals or researchers that are authorized to work and enhance the open source scanners with acceptance of the owner. Next to this process, the open source and academic Web penetration testing scanners are also known as commercial Scanners, like AppScan, Acunetix, Bugblast, Netsparker, etc. These commercial Scanners are basically dis-similar from academic and open source Scanners in such ways that the person can only use these functionalities of the defined scanners by means of purchase, and also the architecture, algorithms or methodologies recycled by development of these scanners. These are not obtainable to public and no other vendor allows enhancement of their scanner [15]. Scanner provide to user with great and vast aid and functionalities factor that are not present in academic and commercial Scanners. Primarily, there are two most standard methods that use in evolution of web penetration testing scanner, whether it is academic scanners, open source or commercial or combination of both. These techniques and methodologies that can be categories into two: static and dynamic techniques. Scanners that perform dynamic approach are basically known as attacks scanners because they explore server response to find vulnerabilities with the help of target attacking application. Moreover, they do not require target source code to execute the security outcomes. Furthermore, these scanners are helpful in static approach that demand to discover the source code of target application and recognize errors or vulnerabilities through flow control of data and information, taint exploration, modelling checking and applying more with the help of above composition [16]-[18]. Jovanovic et al. suggested Pixy, a fixed code investigates the scanner kit that is useful to discover the taint-style of vulnerabilities automatically. In that technique, there are inter procedural, flow and it shows sensitivity in content of low and false attributes and higher accuracy. Mathematical results explain that Pixy was easily find out the both Structured Query Language (SQL) injection and Cross Site Scripting (XSS) vulnerabilities in PHP scripts, these vulnerabilities have an observation about 50% false positive rate [19]. However, WUM puts a greater focus on the technique proposed in [6], [13].

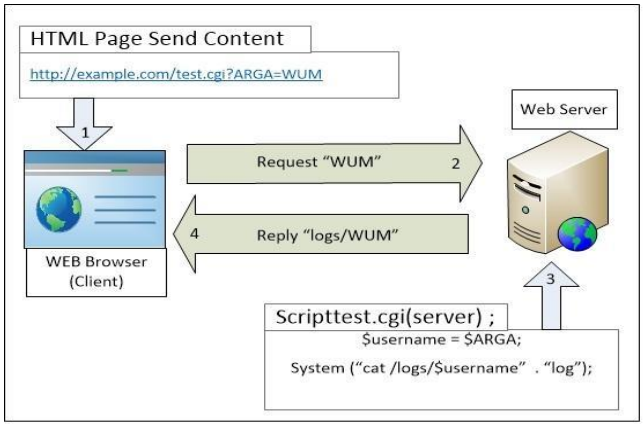

Fig. 2. A simple attack.

Take an explanation that shows the issue with the help of little description of simple example. Fig. 2 expresses the Mentioned scenario. Those web browsers, where a HTML page is accomplished calls a server that uses ARGA as argument. This argument might be explained at the HTML page with the help of defined source code:

```
<form a c t i o n =" t e s t. c g i" method=GET>
<i n p u t maxlength =11 t y p e =" i n p u t" name=
"ARGA"> Username </ i n p u t ></form >
```

Following above source code showed a maximum length and a type that assigned to ARGA. It is defined in such a way that when the server acquires the request that are coming from the web page as a test.cgi file with the content represented in Fig. 2 is achieved. There are different types of vulnerabilities and these are explained further in following sub-section.

A. SQL Injection Vulnerability

Structured Query Language SQL is a database text language that allows user to manipulating the data saved in the database through the commands such as UPDATE, DELETE and INSERT [20]. The major consequences of SQL injection one type of security Exploit are SQL Injection that contains all the SQL queries that can be executed without any appropriate validation format. The common alternative way is that when a mischievous end user inputs certain type of data by which an application uses as shown in Fig. 3 [12].

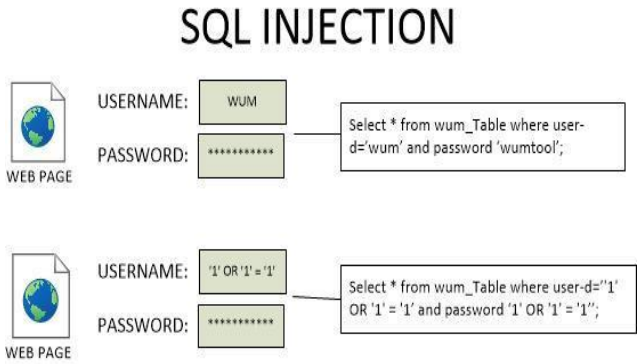

Fig. 3. A SQL injection attack.

B. Cross Site Scripting Vulnerability

Cross site scripting vulnerabilities are namely document object model (DOM), reflected or non-persistent and persistent [21] Cross-site scripting (XSS) vulnerability [22] that allows mischievous developers that they can forward some harmful JavaScript to the site. It can occur, when an application takes the information which is send by user in response pages without doing any validation inspection, while the end user is trying to enter in an injected area of website. Thus, the browser easily achieves his target and mischievous user easily adds malicious JavaScript. Cross site scripting vulnerability arouse when given input is not appropriate. Input sanitization and validation help removing XSS ensuring that given data is in appropriate format of web application as shown in Fig. 4 [22].

Stored or persistent vulnerability is that when hacker inputs pay-loads are stored in web database in a server and that stored data is hacked by post response page. Previous researches observed this kind of bugs on blogs, social media and forums [22]-[24]. DOM-based XSS exposure is occurred when hacker entered in client site and used his JavaScript. Payload inserts in the website and drastically they achieved response from the DOM, these types of attacks are basically done in client site [24].

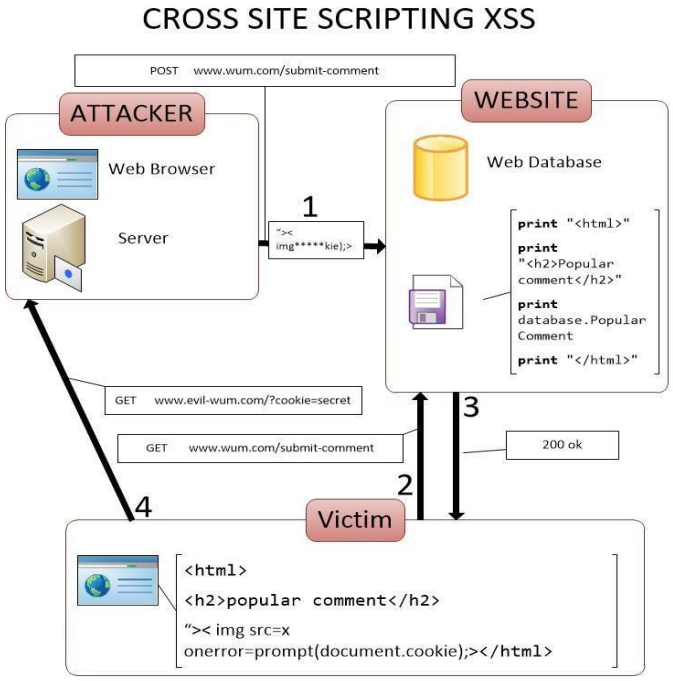

Fig. 4. A XSS attack.

C. Cross Site Request Forgery (CSRF) Vulnerability

CSRF attacks that allows the hacker to do unwanted action in website, blog and emails by launching HTTP Request from browsers. CSRF attacks and severity of the damage in a term money and confidential data and performed through different request as presented in Fig. 5. They can easily change user accounts details, email and password and even performing illegal financial transactions and so on [25].

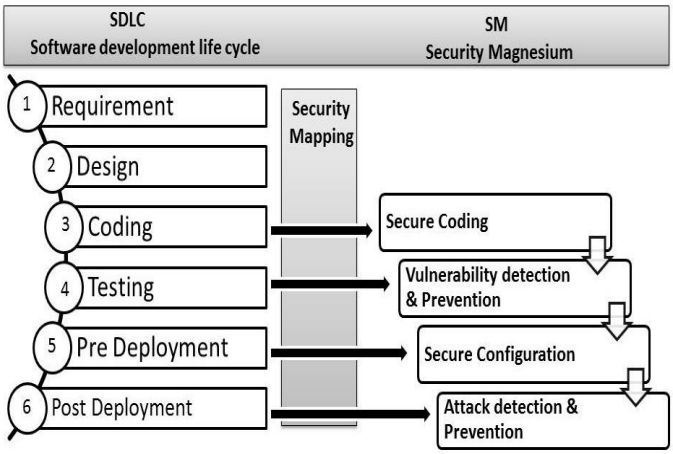

Fig. 5. A CSRF attack.

There are two best important approaches to find vulnerabilities in web applications:

D. Penetration Testing Techniques

1) White box testing: White box testing is an investigation of web application source code. Pixy and FORTIFY scanner easily achieve this task automatically or can be achieved manually. The complexity of a source code can cause problem in accomplishing the task [26].

2) Black box testing: Another methodology for analyzing the security issues in applications is to check the output of the application by providing some input to that specific output. This approach can be easily performed on extensive path through a variety of applications. This tool does not identify the internals of the web application and BBT uses fuzzing technique over the web HTTP requests [26]. the black box methodologies commonly have less incorrect positives than white-box methodologies.

The essential requirement is to go on a page that is able to catch the susceptibilities. Traditional black box web scanners drag a website to count all accessible web pages and formerly down the input records like form values, URL constants, and cookies to generate web vulnerabilities. Though, this method does not pay any attention towards the key points of modern web applications. The state of web application is easily change by providing any application or request. One of the most common situations is that the information of any web application such as: database, file system and time etc. is able to check its output details.

E. Classification of Web Application Security Approaches

Web Application security is a process of engineering the web application by attacking. The malicious attack in web application security works itself to protect the site. The main objective of security is to evade vulnerabilities in the starting phases of development of web life cycle. This life cycle methodology should be followed to make sure completeness and consistency of project. It includes planning, analysis, design and development, testing, and implementation and maintenance as shown in Fig. 6. Following methodologies can be considered in protective programming (i.e. protected coding strategies), detection of vulnerabilities methods and prevention of attack methods.

1) Secure Coding Guidelines: This methodology should be adopted by the developers to make a secure web application. In order to perform this methodology the developers should be trained to learn standards of coding in detail because most of the web vulnerabilities like SQLi and XSS arise due to the incorrect use of inputs. In order to eliminate this type of attack secure coding guide line is the best approach to use But still few developers exist who do not use secure coding standard and make mistakes in their codes through which secure coding guidelines does not promised the security of application [1].

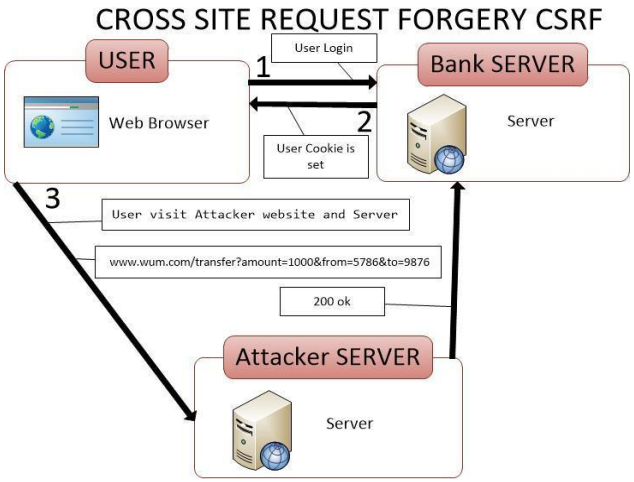

Fig. 6. Process of SDLC.

2) Vulnerability Detection Approaches: VDA is used to detect any type of web vulnerability in the website through performed testing in web application. There are some different approaches to detect web vulnerabilities which are categorized into static, dynamic and hybrid analysis. Static approach is examining web source code without executing website. On the other hand, dynamic is used to detect web vulnerabilities after executing the code. Static and dynamic is used in coding or in testing stage of development life cycle of application. Code-based approach is applied on static method to abstract the valid and invalid situations in code of application. Overall, the value of code based build upon the test cases which was used for identifying vulnerabilities in code. The positive side of static is that they evaluate the code automatically during the early development of life cycle. By doing this, it will be helpful for finding and eliminate errors in early stage and decrease cost, because cost rises along with the development of life cycle. On the other hand, they sometimes generate wrong results i.e. producing a wide range of false positives and false negatives. Somehow, dynamic approach is generating true result. They need a massive amount of test cases to detect errors in code [1].

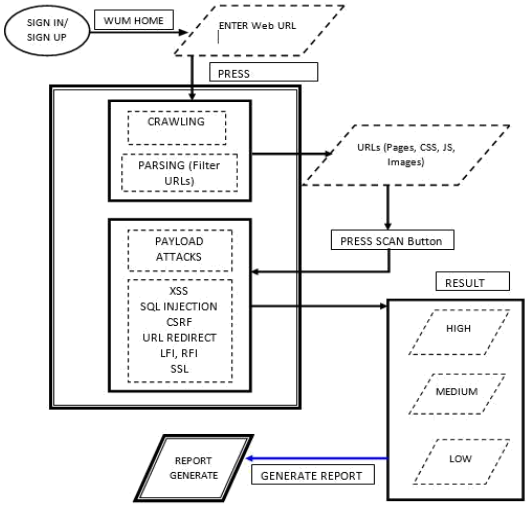

Fig. 7. The Architecture of WUM Tool.

III. Methodology

The proposed method is based on WEB security WUM scanner in order to find web vulnerabilities. It's also display parameters from where SQLi, XSS and some well-known vulnerabilities were founded. In order to keep the design open, collective and compatible architecture has been used. This scanner consists of four parts named crawling and parsing, detection, and attacking and analysis phases. It's also providing detail on web vulnerability in last to show generated attack part that can be trigged separately. As far as the efficiency and performance is concerned, this scanner is able to dispatch 8 to 10 parallel attacks that is further elaborated in given Fig. 7.

IV. Automated Vulnerability Detection

An Automated web vulnerability scanner to find efficient result approach depends upon imitation of SQLI, XSS, CSRF, LFI/RFI vulnerabilities payload. Consequently, the possibility of examining is restricted only to HTTP responses received from the application server which runs verified web application. Likewise, to the reported strategies commonly found in other systems [6], [11], [14]. Our approach encompasses the following Modules. Web crawling, AEP's (application entry Point's) detection and extraction, attacking, analysis, and report generation as shown in Fig. 8.

1) Crawling Module: Attacks can be propelled just against formerly recognized AEP's during the dynamic security analysis. Therefore, identification of all pages inside target web application is critical for testing [14]. This can be done automated, manually or semi-automated. It will crawl page to page of website and will automatically check the pages with scripts and payloads for vulnerabilities. Comparatively slow response time of remote website server. To initiate a crawling session, the crawling phase of scanner needs to be linked with a website URL. Crawler use URL as a starting point and steps down to the web link tree and collecting all web pages associated with it. Only as a specific web crawler, this scanner has configurable options for the maximum web pages per domain to crawl high web pages depth as shown in Fig. 9. The basic idea for implementation of crawling component was taken from existing systems [9], [11], [14], [27], [28].

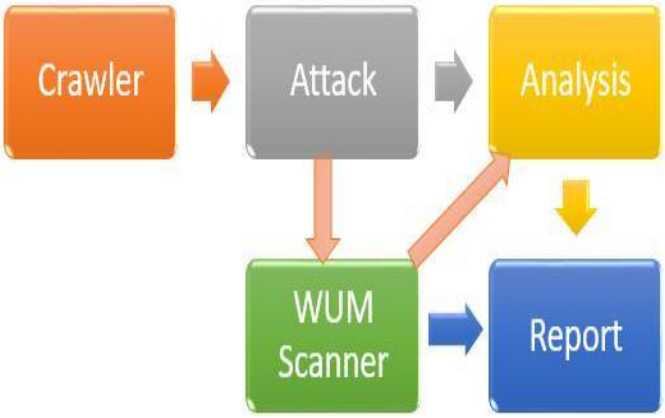

Fig. 8. An overview of a WUM.

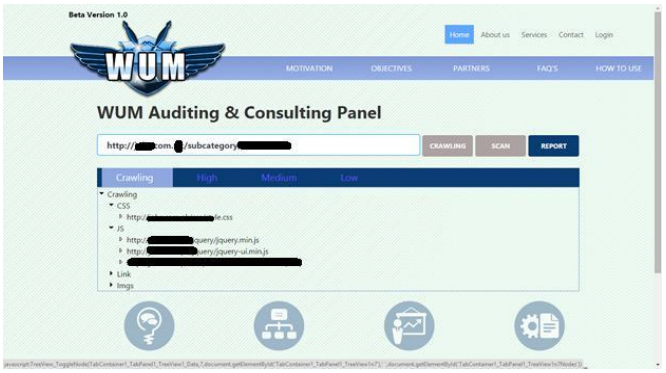

Fig. 9. WUM crawling phase.

2) Attack Module: Once crawling phase was completed; next phase is to initialize processing on the list of target web pages. Particularly, the attack module scans each page which found in a crawling phase as presented in Fig. 10. For each AEP, a set of valid parameter values is generated that are used by different researchers [9], [11], 14] to generate HTTP request. The outcome of this request is referenced in HTML page. In addition, in every AEP, a set of malicious or incorrect parameter values is created. Furthermore, parameter values which violate predefined constraints of parameters are generated.

3) Analysis Modules: The third module is analysis module. When user click on more details it launches attack to interpret web pages and parse web vulnerabilities There are some possibilities of false positives web vulnerabilities. To reduce this problem, a module is added on WUM to care about confidence value in case of false positives are occurred. At the end of this phase scanner is able to provide solution of web vulnerabilities which was found by WUM scanner.

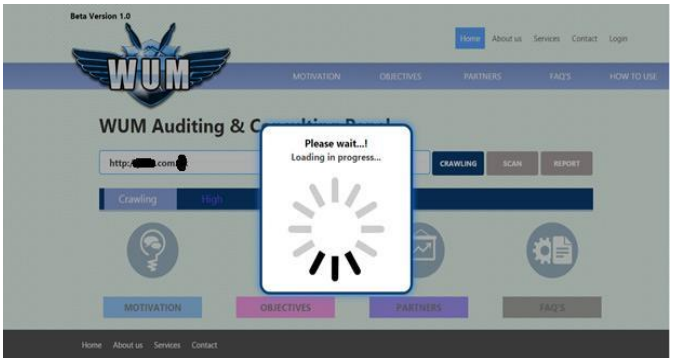

Fig. 10. WUM analysis phase.

V. Experimental Result

WUM scanner has been developed in ASP.net with the help of the database named Microsoft SQL Server. For our studies, we use sample size of 10 website which was selected from malicious website obtained from XSSed (xssed.com) and DMOZ (dvwa.co.uk).in order to test different scanners, for presented study select Sample web URL used as AEP on WUM scanners: scanner 1, scanner 2 and scanner 3. Some of them are able to find all web vulnerabilities as wum scanner. We also selected some well-known vulnerabilities, like XSS,

SQL Injections, CSRF, LFI/RFI, SSL and URL redirection. During testing we have used 1 commercial and 2 open source web vulnerability scanners to compare results with WUM scanner.

The result of different scanners presented in Table I. In order to evaluate result for presented we performed testing on different scanner to find web vulnerabilities. Many attempts have been made in order to aim to find web vulnerabilities on sample websites for the current study present interesting result from our Dataset. From the Table scanner 1 have found all web vulnerabilities expect SSL because this scanner is not able to find SSL web Vulnerability. For this study we have marked SSL as NA. It's found total 27 vulnerabilities in case of XSS is 7, in case of SQL Injection is 6, in case of CSRF is 3, in case of LFI/RFI is 1, in case of CJ is 6 and in case of directory discloser is 4. In the same way Scanner 2 and 3 found different result display with total 17 and 9 vulnerabilities. WUM scanner found better result as compared to others scanner with total 38 vulnerabilities. Its founds result in case of XSS is 9,in case of SQL Injection is 5, in case of CSRF is 4, in case of LFI/RFI is 2, in case of CJ is 7, in case of SSL is 8, and in case of directory discloser is 3. This study presents interesting result to detect web vulnerabilities with respect to exiting result.

Table II define Resampling of scanner vulnerabilities comparison and contains the means values of vulnerabilities and accuracy percentile over sample data. As compared to others WUM scanner generated a mean value of 0.54 which is more precise then other scanners. These results are driven from mean values and it is clearly obvious that WUM scanner has a competitive advantage over the rest of tools.

TABLE I. SCANNER VULNERABILITIES COMPARISON

Scanners	XSS	SQLi	CSRF	LFI/RFI	CJ	SSL	DD	Total
Scanner 1	7	6	3	1	6	NA	4	27
Scanner 2	5	4	6	2	NA	NA	NA	17
Scanner 3	6	3	NA	NA	NA	NA	NA	9
WUM	9	5	4	2	7	8	3	38

TABLE II. RESAMPLING OF SCANNER VULNERABILITIES COMPARISON AND ACCURACY PERCENTILE

TOOL	XSS	SQLi	CSRF	LFI/RFI	CJ	SSL	DD	Mean	%
S1	0.70	0.60	0.30	0.10	0.60	NA	0.40	0.45	45%
S2	0.50	0.40	0.60	0.20	NA	NA	NA	0.425	42%
S3	0.60	0.30	NA	NA	NA	NA	NA	0.45	45%
WUM	0.90	0.50	0.40	0.20	0.70	0.80	0.30	0.5428	54%

The above result presented clearly shows that WUM scanners have 54% accuracy ratio and Scanner 1, Scanner 2 Scanner 3 have 42%, 45% and 45% accuracy ratio respectively. Our scanner is more precise and has increased accuracy result by 9% with comparison to scanner 1, in case of Scanner 2 increased accuracy result by 12%, in case of Scanner 3 increased accuracy result by 9%. This comparison of different scanner with WUM is also presented in Fig. 11.

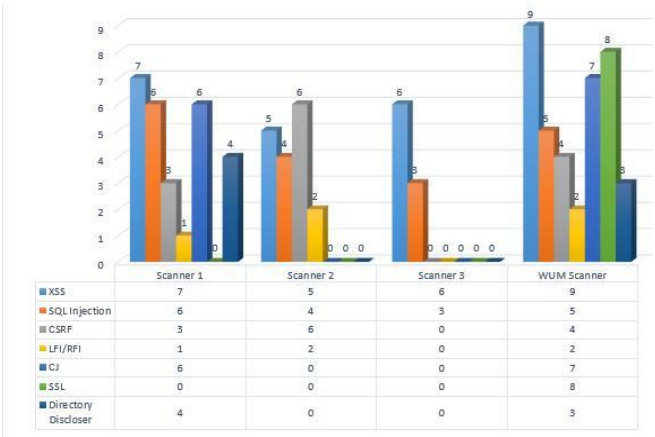

Fig. 11. Scanner vulnerabilities comparison.

VI. CONCLUSION

In this study, we tried to enlighten the most common vulnerabilities of websites, such as Cross-Site Scripting, SQL Injection, Cross site request forgery CSRF, LFI/RFI, CI, SSL, DD. Additionally, we have developed a new scanning tool i.e. Website Unique Method (WUM) to detect these vulnerabilities. To provide factual results, the experimental work is carried out on proposed vulnerability scanning tool along with other well-known scanners is tested on 10 malicious websites to demonstrate the viability and the effectiveness of the proposed solution. The experiments include the evaluation of detection rate of vulnerability scanning system for XSS, SQLi, CSRF, LFI/RFI, CI, SSL, DD and the assessment of the effectiveness of our proposed methodology. The experimental results show that the proposed approach effectively detects most of the vulnerabilities. Moreover, the proposed approach allows a website developer's to recognize and assess vulnerabilities prior to publish their websites on web.

Future research will be based on the development of an upgraded version of WUM scanner to prevent and detect more web attacks, parameters and payloads to test random attacks by using all permutation and combinations. We are planning to implement machine learning on these set to identify more efficient result. We are also setting up a WUM website for users to scan website and download scanner.

REFERENCES

[1] Deepa, G., and P. Santhi Thilagam. "Securing web applications from injection and logic vulnerabilities: Approaches and challenges." Information and Software Technology 74 (2016): pp. 160-180.

[2] Kaur, Daljit, and Parminder Kaur. "Empirical analysis of web attacks." Procedia Computer Science 78 (2016): pp. 298-306.

[3] Gupta, Mukesh Kumar, Mahesh Chandra Govil, and Girdhari Singh. "Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications." In Computer Science and Software Engineering (JCSSE), 2015 12th International Joint Conference on, pp. 162-167. IEEE, 2015.

[4] Awoleye, Olusesan M., Blessing Ojuloge, and Mathew O. Ilori. "Web application vulnerability assessment and policy direction towards a secure smart government." Government Information Quarterly 31 (2014): pp.S118- S125.

[5] OWASP: Available at http://www.owasp.org/index.php/Category: OWASP Top Ten Project, 2017. Retrieved 24/09/2017.

[6] Bozic, Josip, and Franz Wotawa. "PURITY: a Planning-based secURITY testing tool." In Software Quality, Reliability and Security-Companion (QRS-C), 2015 IEEE International Conference on, pp. 46-55. IEEE, 2015.

[7] Antunes, Nuno, and Marco Vieira. "Benchmarking vulnerability detection tools for web services." In Web Services (ICWS), 2010 IEEE International Conference on, pp. 203-210. IEEE, 2010.

[8] Livshits, Benjamin V., and Monica S. Lam."Finding security errors in java programs with static analysis (tech report)." (2005).

[9] Djuric, Zoran. "A black-box testing tool for detecting SQL injection vulnerabilities." In Informatics and Applications (ICIA), 2013 Second International Conference on, pp. 216-221. IEEE, 2013.

[10] Liban, Abdilahi, and Shadi MS Hilles. "Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack." In Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th, pp. 47-52. IEEE, 2014.

[11] Kals, Stefan, Engin Kirda, Christopher Kruegel, and Nenad Jovanovic. "Secubat: a web vulnerability scanner." In Proceedings of the 15th international conference on World Wide Web, pp. 247-256. ACM, 2006.

[12] Doupe,´ Adam, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna. "Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner." In USENIX Security Symposium, vol. 14. 2012.

[13] Halfond, William GJ, and Alessandro Orso. "AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks." In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, pp. 174-183. ACM, 2005.

[14] Huang, Yao-Wen, Shih-Kun Huang, Tsung-Po Lin, and Chung-Hung Tsai. "Web application security assessment by fault injection and behavior monitoring." In Proceedings of the 12th international conference on World Wide Web, pp. 148-159. ACM, 2003.

[15] OWSAP Open Web Security Project Available at https: //www.owasp.org/index.php/Category:VulnerabilitLScanning Tools, Retrieved 24/09/2017.

[16] Zhang, Xin-hua, and Zhi-jian Wang. "Notice of retraction a static analysis tool for detecting web application injection vulnerabilities for asp program." In e-Business and Information System Security (EBISS), 2010 2nd International Conference on, pp. 1-5. IEEE, 2010.

[17] Zhang, Lijiu, Qing Gu, Shushen Peng, Xiang Chen, Haigang Zhao, and Daoxu Chen. "D-WAV: A web application vulnerabilities detection tool using Characteristics of Web Forms." In Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, pp. 501-507. IEEE, 2010.

[18] Fonseca, Jose, Nuno Seixas, Marco Vieira, and Henrique Madeira. "Analysis of field data on web security vulnerabilities." IEEE transactions on dependable and secure computing 11, no. 2 (2014): 89-100.

[19] Jovanovic, Nenad, Christopher Kruegel, and Engin Kirda. "Pixy: A static analysis tool for detecting web application vulnerabilities." In Security and Privacy, 2006 IEEE Symposium on, pp. 6-pp. IEEE, 2006.

[20] Singh, Nanhay, Mohit Dayal, R. S. Raw, and Suresh Kumar. "SQL injection: Types, methodology, attack queries and prevention." In Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, pp. 2872-2876. IEEE, 2016.

[21] Wang, Ran, Guangquan Xu, Xianjiao Zeng, Xiaohong Li, and Zhiyong Feng. "TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting." Journal of Parallel and Distributed Computing (2017).

[22] Shar, Lwin Khin, and Hee Beng Kuan Tan. "Defending against cross-site scripting attacks." Computer 45, no. 3 (2012): 55-62.

[23] Hydara, Isatou, Abu Bakar Md Sultan, Hazura Zulzalil, and Novia Admodisastro. "Current state of research on cross-site scripting (XSS)–A systematic literature review." Information and Software Technology 58 (2015): 170-186.

[24] Gupta, Shashank, and B. B. Gupta. "Enhanced XSS Defensive Framework for Web Applications Deployed in the Virtual Machines of Cloud Computing Environment." Procedia Technology 24 (2016): 1595-1602.

[25] Shahriar, Hossain, and Mohammad Zulkernine. "Client-side detection of cross-site request forgery attacks." In Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on, pp. 358-367. IEEE, 2010.

[26] Vieira, Marco, Nuno Antunes, and Henrique Madeira. "Using web security scanners to detect vulnerabilities in web services." In Dependable Systems Networks, 2009. DSN'09. IEEE/IFIP International Conference on, pp. 566-571. IEEE, 2009.

[27] David Cruwys. C Sharp/VB - Automated WebSpider/ WebRobot Available at: https://www.codeproject.com/Articles/6438/ C-VB-Automated-WebSpider-WebRot, Retrieved 24/09/2017.

[28] Iqbal, Muhammad, Malik Muneeb Abid, Usman Waheed, and Syed Hasnain Alam Kazmi. "Classification of Malicious Web Pages through a J48 Decision Tree, aNa¨ıve Bayes, a RBF Network and a Random Forest Classifier for WebSpam Detection." JUSTNESS vol. 10, No. 4, pp. 51-72, April 2017