ISSN 2278-3091
Volume 9, No.1.3, 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse6991.32020.pdf
https://doi.org/10.30534/ijatcse/2020/6991.32020

Web Vulnerability Assessment Tool for Content Management System

Mohamad Yusof Darus[1], Mohd Afham Omar[2], Mohd Farihan Mohamad[3]
Zulhairi Seman[4], Norkhusahini Awang[5]
[1]Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia, yusof@fskm.uitm.edu.my
[2]Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia, angah_iezafham89@yahoo.com
[3]Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia, mohammadfarihanmohamad@gmail.com
[4]Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia, hello@zulhairiseman.net
[5]Universiti Teknologi MARA, 40450, Shah Alam, Selangor, Malaysia, shaini@fskm.uitm.edu.my

ABSTRACT

Using web application through internet communication has become an important part of our lives. Every day we interact with a large number of custom-built web applications that been implemented using different technologies. Prior to the introduction of web standards, developers faced the problem of browser incompatibility. The highly heterogeneous nature of the web with its different implementation languages, encoding standards, browsers and scripting environments makes it difficult for web application developers to properly secure their applications and stay up-to-date with emerging threats and newly discovered attacks. Web vulnerabilities assessment tools have become a necessity due to the increased security threats posed by hackers and other criminals who are on the lookout for confidential information. In this research, we proposed web assessment tool called SNEAKERZ that detected vulnerability and automatically analyzed exploitable web vulnerabilities and proposed solutions for that vulnerability. Our proposed solutions looked for security loopholes in web applications to prevent hackers from gaining unauthorized access to corporate information and data.

Key words: Web vulnerability, vulnerability assessment, web security, software vulnerability, content management system

1. INTRODUCTION

Web security should be a priority in every organization. Vulnerability attack in a threat that can ruin the company's reputation. Furthermore, 24/7 accessibility to the systems, insecure web applications offered access to backend of databases and allow hackers to perform illegal activities. For example, there has been extensive press coverage of recent security incidences involving the loss of sensitive data belonging to millions of customers. According to [1] stated that by transmitting data processes, such as website forms, emails, instant messaging and file transfer which are unmonitored and unregulated to their destination could make data leakage issue become intensifies. This situation explain how data processing through browser exposed to web vulnerability. In this research, we developed web assessment tool that detected vulnerabilities in different types of content management system and analyze the vulnerabilities based on the report generated by the proposed web vulnerabilities assessment tool. The proposed tool has ability to automatically discover and exploit application-level web vulnerabilities in a large number of web applications and automatically analyzes exploitable web vulnerabilities. Furthermore, SNEAKERZ listed all web vulnerabilities and proposed solutions from the vulnerability detected. We focus on five types of vulnerabilities, which are SQL Injection, Cross-Site Scripting (XSS), Carriage Return and Line Feed (CRLF) Injection, Remote File Inclusion (RFI) and Local File Inclusion (LFI). The purposes of SNEAKERZ are to identify the weaknesses of web application, detect known vulnerabilities and security weaknesses. From the research [2] mention that SQL injection, cross site scripting (XSS) and remote code execution are common attacks that can disable web services and steal sensitive user information. A discussion from [3] mention that the attacker use web loophole to achieve cross-site browsing for session stealing, legitimate user masquerading or credential information stealing. Another study from [4] stated that vulnerability to a web application can cost financial loss, impersonate users in social platforms or reveal our personal information. Authors also mention that web application attacks are the most common cause of data breach today.

The security loophole in web applications is being exploited by hackers worldwide. According to a survey by the Gartner Group, almost three-fourths of all Internet assaults are targeted at web applications. The Open Web Application Security Project (OWASP) Foundation was listed the top 10 of web application vulnerabilities and security risks. From [5] mention that these vulnerabilities were found in public API interfaces which are 1) Cross Site Scripting (XSS), 2)

Injection Flaws, 3) Information Leakage and Improper Error Handling, 4) Broken Authentication and Session Management, 5) Insecure Cryptographic Storage, 6) Insecure Communications, and 7) Failure to Restrict URL Access. It is needed to have a platform to do the web vulnerabilities assessment since it is causing an information losses and company reputation. A study from [6] stated that current state of research in the security shown that 55.56% of the studies focus on web service attacks and 16.67% on vulnerabilities. A few researcher looked at the web vulnerabilities compare to web services. However, web vulnerabilities exposed to data breaches. Another researcher also mention that organization which expose to web vulnerabilities give an impact on technical assets or business including privacy violation, financial damage, damage to the reputation of companies, noncompliance, loss Integrity, accountability, and availability [7]. It is important to study how to prevent web vulnerabilities by doing the web assessment.

The web vulnerability assessment is a process of identifying, quantifying and prioritizing the vulnerabilities in web system. This process uses certain ways and tools to find vulnerabilities. According to [8] stated that there are three types of defensive coding practices for SQL injection prevention which are input type checking, input encoding and positive pattern matching. It usually is a part of web risks assessments during the project definition stage or as a result of penetration testing of an existing website or web application. Risks assessment is a detailed and thorough identification of the assets and security threats. Authors also recommend to do a proactive assessment, including traditional black box testing on the servers. Another research from [9] mention that the most important parameters that can be used as a guidelines for identifying and selecting appropriate vulnerability assessment tools are 1) open source or freeware, 2) TLS/SSL protocol, 3) HTTP proxy, 4) shared hosting, 5) outdated server component, 6) security misconfigurations, and 7) host authentication & user account enumeration. Refer to Table 1, authors [10] mention about number and proportion of vulnerabilities associated with a specific keyword for the labeled vulnerabilities. From the table we can conclude that web browser exposed to vulnerabilities since content management system used web browser to use the system.

Table 1: Number of Vulnerabilities Per Type and Web Browser[10]

Keywords	Explorer	Safari	Firefox	Chrome
Denial of service	90 (36.3%)	162 (77.1%)	354 (49.2%)	634 (79.65%)
Execute code	114 (46.0%)	150 (71.4%)	432 (60.0%)	53 (6.7%)
Overflow	41 (16.5%)	98 (46.7%)	158 (21.9%)	203 (25.5%)
SQL injection	0	0	0	0
Obtain information	18 (7.3%)	17 (8.1%)	63 (8.75%)	36 (4.5%)
Gain privileges	0	0	15 (2.1%)	0
Bypass restriction or similar	32 (12.9%)	19 (9.05%)	95 (13.2%)	80 (10.05%)
Directory traversal	5 (2.0%)	2 (0.95%)	6 (0.8%)	2 (0.25%)
Cross site scripting	23 (9.3%)	10 (4.8%)	75 (10.4%)	29 (3.6%)
Http response splitting	1 (0.4%)	0	1 (0.1%)	0
CSRF	0	0	7 (1.0%)	3 (0.4%)
Memory corruption	33 (13.3%)	130 (61.9%)	220 (30.6%)	59 (7.4%)

Conducting a vulnerabilities assessment is to prevent system from threats and data leakage. According to [11] mention that web applications which interact with back-end database. To complete the transaction, web applications often interact with back-end components and have access to sensitive user information, such as passwords, credit card numbers or user personal information. These become reasons an attractive target for attackers to steal the information.

2. SOFTWARE VULNERABILITY TAXONOMY

In computer security, web vulnerability is the weakness which allows an attacker to reduce a system's information assurance on the website. Vulnerability is the intersection of three elements consists of a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw. When computer is connected to an unsecured network, software security could be compromised. Critical errors in users' computer software can leave data in the entire network vulnerable to a number of malicious threats, including malware, phishing, proxies, spyware, adware, botnets or spam. In this research, we studied on web vulnerability assessment and developed web vulnerability scanner as our project. Before we discuss further on the development of our project, we present the taxonomy of software vulnerability. There are 3 categories that contribute to software vulnerability which are software defect, software bug and software error. The discussion about these categories as mention below.

3.1 Software Defect

Software defect can be define as deviation or irregularity from the specifications mentioned in the product functional specification. It is basically caused by the developers' mistakes. Defect in a software product represents the inability and inefficiency of the software to meet the specified requirements and criteria and subsequently prevent the software application to perform the expected and desired working. According to [12] stated that software defect is the root cause of software failure and affects software reliability. Other researcher also explained that software defect have a large impact on software quality such that can increase software maintenance cost [13]. The same author also mentions about their research on software defect scoping to defect management, defect analysis and defect analysis which these lead to software testing. From [14] mention that the cost of defect correction is significantly high after software testing. For open source software, software reliability also plays an important characteristics of software quality when considered for commercial use. From [15] mention that software defect affects the functionality of a given specification to the software system. From their studies shown the open source software appear to be unstable in the area of low-level design.

However, open source does not suffer major missing function and code issues compared to in-house developed software.

3.2 Software Bug

Software bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. From [16] define that software bugs are from human error and are often attribute to poor understanding, inexperience, lack of skill and incompetence. Study from [17] mention that 3 categories in representing reproducibly null software bugs are reproducibility, extensibility and usability. The discussion on reproducibility mention that program behavior is often dependent upon the exact configuration of its host environment at both compile- and run-time such as library and compiler versions, environment variables. Next, extensibility is when programmer should be able to reproducibly evaluate new techniques on the datasets used in previous evaluations, and reliably extend those datasets. Last, usability is when environments should be lightweight, both in terms of disk space and their run-time overhead and should be easy to install and use. In other research, [18] mention that code smell are more likely to contain software bugs. They also stated that 53% of the software problems are not directly related to code as follows: lack of adequate technical infrastructure, developer's coding habits, external services such as web services, run-time environment and defects initially present in the system. The remaining, from the 47% associated with Java source code, up to 27% was related to code smells.

3.3 Software Error

Software error is a mistake, misconception, or misunderstanding on the part of a software developer. Software developer include software engineers, programmers, analysts, and testers. From [16] define that software error happen even though programmers have planned sequence of mental or physical activities but fails to achieve its intended outcome. Authors also stated that software error associated with software bugs. Software error arise when behavior is specified in tests, while classes are implemented in periods when functionality is introduced and modified. Another research on software error mention that the important of balance between managerial and technical perspectives [19]. They also stated that software error can be viewed as a managerial issue at the organization level. According to [20] stated that human errors are the primary cause of software error since computer programs are a pure cognitive product that describes its designers' thoughts. From their finding also stated that software developers tend to commit post-completion error with a high likelihood once the post-completion scenario is presented in software requirements.

3. SNEAKERZ DEVELOPMENT

In this sub-topic, we discussed the development of SNEAKERZ as a web vulnerability assessment tool. As mentioned, the proposed tool divided into three (3) main modules:

a. Interface and Integration - Python language is used as a core programming language, which can detect type of websites such as Drupal Content Management System (CMS), Joomla Content Management System (CMS), Wordpress Content Management System (CMS) and normal website. Integration phase is to integrate the interface and web vulnerability detection engine.

b. Web Vulnerability Detection - We focused on finding five types of vulnerabilities.

c. Generate Report - The report shown a list of all web vulnerabilities and proposed solutions for admins. To generate the report, this project used combination of programming language such as XML language, HTML language and Python language. A few websites will be used in the process of testing.

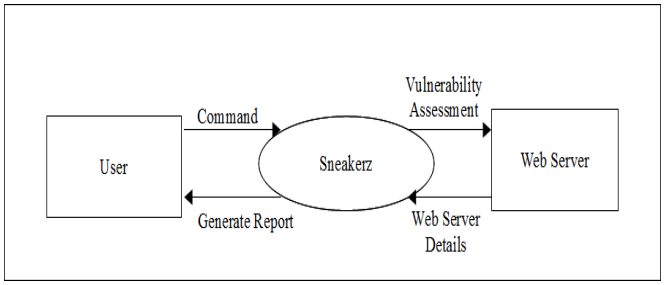

Figure 1: SNEAKERZ Flow Diagram

As shown in Figure 1, the proposed SNEAKERZ as a web vulnerability scanner tool scanned the targeted URL addresses is show in the flow diagram. It scans type of website based on CMS type. SNEAKERZ list all the vulnerability and also the possible solutions. The information then saved in log file as a report.

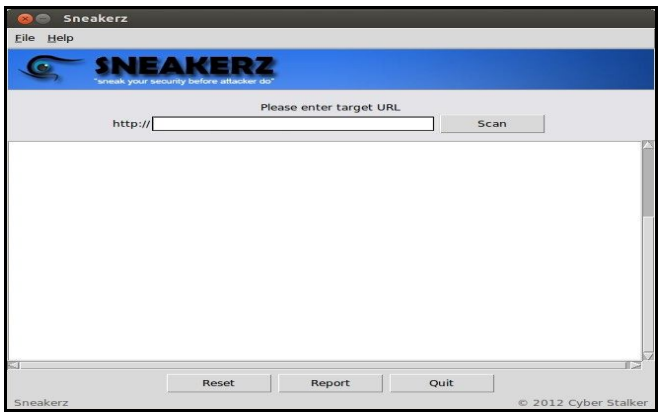

Figure 2: SNEAKERZ Main Menu

Figure 2 shows the interface of the proposed SNEAKERZ web vulnerability assessment tool. The SNEAKERZ is developed to test on vulnerable websites. For this research, we tested some of the CMS which have vulnerability as our testbed. The CMS that we tested are:

1. geek.zulhairiseman.net (Drupal Content Management System)
2. www.redahmalaya.com (Joomla Content Management System)
3. life.zulhairiseman.net (Wordpress Content Management System)
4. www.dulcemarialive.mx (Normal Website)

Figure 3 shows the SNEAKERZ process engine in flow diagram. From the diagram explained how SNEAKERZ accepted URL and process to choose type of CMS to do the analysis type of vulnerability from the CMS page.

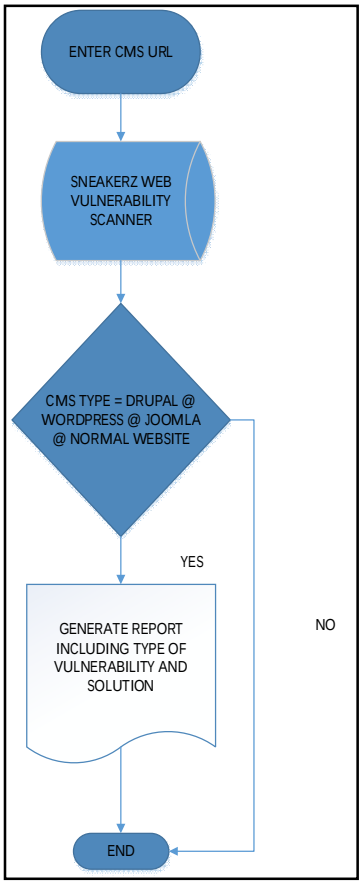

Figure 3: SNEAKERZ Process Engine

4. RESULTS

From Figure 4 shows example report generated by SNEAKERZ from Drupal CMS. In the generated report list all web vulnerabilities on website from one of the vulnerability CMS which is *geek.zulhairiseman.net*. SNEAKERZ engine launched web vulnerability detection module. There are seven detection modules which are SQL module, File Handling module, XSS module, CRLF module, Execution Command module, Blind SQL module and Permanent XSS module. Types of web vulnerability that SNEAKERZ scanned are 1) SQL Injection, 2) Cross-Site Scripting, 3) CRLF Injection, 4) Local File Inclusion and 5) Remote File Inclusion. SNEAKERZ suggested the solution on preventing web vulnerability from happen.

The report generated in SNEAKERZ indicated the vulnerability level on the website which from this experiment shown three risk levels of vulnerability. High risk level vulnerability is a vulnerability that allowed an intruder to immediately gain privileged access to the system. SQL injections, Blind SQL injection, XSS, File Handling, CRLF, and Command Execution were set as high risk level vulnerability. For medium risk level vulnerability when an intruder can access to the system that is not privileged access. Resource consumption was set as medium risk level vulnerability. Lastly, for low risk level vulnerability when intruder can compromise with attempting to the system. Raw XSS was set as low risk level vulnerability.

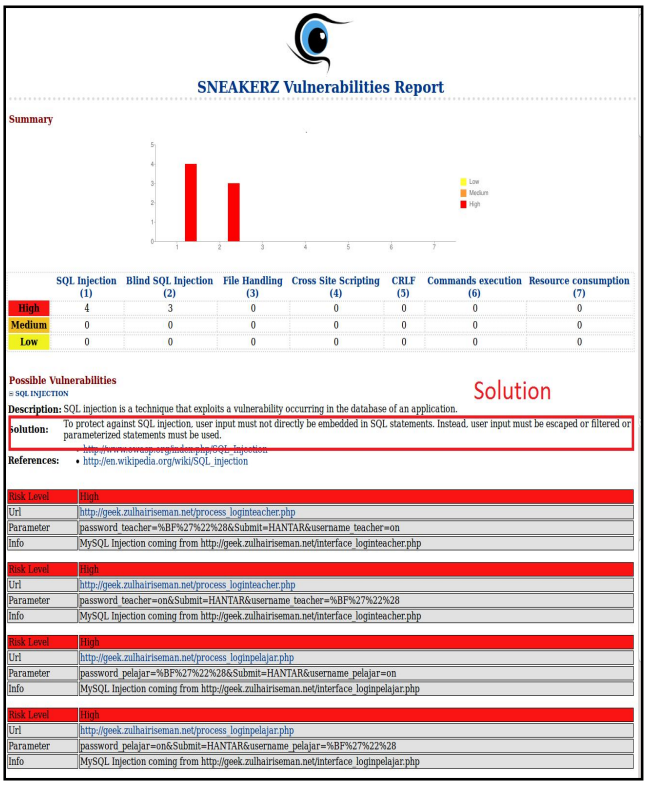

Figure 4: Web Vulnerability's Report for Drupal CMS

5. CONCLUSION

The proposed web vulnerability assessment and scanner called SNEAKERZ is a testbed tool for Microsoft Windows and Linux platform. This tool detects web vulnerability and suggested a solution the web admin. As mention by [21] in their book that software components vulnerabilities can be

classified as software defects and configuration errors. As a web administrator, web application security is crucial, and it is a process of protecting websites and online services against security threats that exploit vulnerabilities in an application's code. Web application vulnerabilities are result from lack of input/output sanitization, which are exploited to either manipulate source code or gain unauthorized access.

ACKNOWLEDGMENT

The authors would like to thank the Universiti Teknologi Mara (UiTM) for sponsoring this research under the LESTARI Grant 600-RMC/LESTARI SDG-T 5/3 (142/2019) of Research Management Centre and Faculty of Computer & Mathematical Sciences, UiTM Shah.

REFERENCES

1. M. A. M. Ariffin, K. A. Rahman, M. Y. Darus, N. Awang, and Z. Kasiran, **Data leakage detection in cloud computing platform**, Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 1.3 S1, pp. 400–408, 2019. https://doi.org/10.30534/ijatcse/2019/7081.32019
2. Y. Pan et al., **Detecting web attacks with end-to-end deep learning,** J. Internet Serv. Appl., vol. 10, no. 1, 2019.
3. Lim Kah Seng, N. Ithnin, and S. Z. M. Shaid, **The Preparation of Cross-site Scripting in Automated Web Application Vulnerability Assessment: The Quantitative Analysis**, Int. J. Adv. Trends Comput. Sci. Eng., vol. 8, no. 3, pp. 195–200, 2019.
4. A. Rahman, B. Ahmed, M. Amjad, and M. S. Siddik, **Analyzing Web Application Vulnerabilities : An Empirical Study on E-Commerce Sector in Bangladesh**, in International Conference on Computing Advancements, 2020, pp. 5–10. https://doi.org/10.1145/3377049.3377107
5. A. Masood, **Cyber Security for Service Oriented Architectures in a Web 2 . 0 World : An Overview of SOA Vulnerabilities in Financial Services**, pp. 1–6, 2013.
6. R. Rojas, A. Muedas, and D. Mauricio, **Security maturity model of web applications for cyber attacks**, in 3rd International Conference on Cryptography, Security and Privacy, 2019, pp. 130–137.
7. P. Touseef et al., **Analysis of automated web application security vulnerabilities testing**, in 3rd International Conference on Future Networks and Distributed Systems, 2019, pp. 1–8. https://doi.org/10.1145/3341325.3342032
8. B. Delamore and R. K. L. Ko, **Escrow: A large-scale web vulnerability assessment tool**, Proc. - 2014 IEEE 13th Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2014, pp. 983–988, 2015.
9. R. Vibhandik and A. K. Bose, **Vulnerability assessment of web applications-a testing approach**, 2015 4th Int. Conf. e-Technologies Networks Dev. ICeND 2015, pp. 16–21, 2015.
10. Y. Movahedi, M. Cukier, A. Andongabo, and I. Gashi, **Cluster-based vulnerability assessment of operating systems and web browsers**, Computing, vol. 101, no. 2, pp. 139–160, 2019.
11. N. F. Awang, A. A. Manaf, and W. S. Zainudin, **A survey on conducting vulnerability assessment in web-based application**, Commun. Comput. Inf. Sci., vol. 488, pp. 459–471, 2014.
12. F. Zeng, A. Chen, and X. Tao, **Study on software reliability design criteria based on defect patterns**, Proc. 2009 8th Int. Conf. Reliab. Maintainab. Safety, ICRMS 2009, pp. 723–727, 2009.
13. J. Gao, L. Zhang, F. Zhao, and Y. Zhai, **Research on software defect classification**, Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019, no. Itnec, pp. 748–754, 2019. https://doi.org/10.1109/ITNEC.2019.8729440
14. K. Punitha and S. Chitra, **Software defect prediction using software metrics - A survey**, 2013 Int. Conf. Inf. Commun. Embed. Syst. ICICES 2013, pp. 555–558, 2013.
15. S. M. Syed-Mohamad and T. McBride, **Reliability growth of open source software using defect analysis**, Proc. - Int. Conf. Comput. Sci. Softw. Eng. CSSE 2008, vol. 2, pp. 662–667, 2008.
16. T. Lopez, M. Petrel, and B. Nuseibehl, **Examining active error in software development**, Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. VL/HCC, vol. 2016-Novem, pp. 152–156, 2016.
17. C. S. Timperley, S. Stepney, and C. Le Goues, **Poster: BugZoo: A platform for studying software bugs**, Proc. - Int. Conf. Softw. Eng., pp. 446–447, 2018.
18. A. S. Cairo, G. de F. Carneiro, and M. P. Monteiro, **The impact of code smells on software bugs: A systematic literature review**, Inf., vol. 9, no. 11, pp. 1–21, 2018.
19. S. El Koutbi, A. Idri, and A. Abran, **Systematic Mapping Study of Dealing with Error in Software Development Effort Estimation**, Proc. - 42nd Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2016, no. 2, pp. 140–147, 2016.
20. F. Huang, **Post-completion error in software development**, Proc. - 9th Int. Work. Coop. Hum. Asp. Softw. Eng. CHASE 2016, pp. 108–113, 2016.
21. E. Bertino, L. D. Martino, F. Paci, and A. C. Squicciarini, **Security for web services and service-oriented architectures**, in Security for Web Services and Service-Oriented Architectures, 2010, pp. 1–226.