iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.35135

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Vulnerability Scanning Technology on Web Applications

Aarushi Dwivedi[1], M. L. Sharma[2], K. C. Tripathi[3]
[1, 2, 3]*Information Technology, Maharaja Agrasen Institute of Technology, Delhi*

Abstract: ***Modern society is far more dependent on web applications than the previous generations. Even though our dependence is increasing rapidly, the security level is far lower than required. To guarantee the security of the data system in the industry and our daily life, it is especially crucial to find out web application security vulnerabilities quickly and accurately. A vulnerability is a state of being unprotected from the prospect of an attack. It permits an attacker to gain a certain level of command of the site, and possibly the hosting server. One such vulnerability is the cross-site scripting vulnerability. In this exposition, a generic vulnerability scanner is proposed which can be customized to find any number of vulnerabilities. The scanner maps out the website and gives a report of all the vulnerabilities. For the purpose of evaluation, it has been customized to find XSS vulnerability in web applications.***
Keywords: Vulnerability scanner, Cross-site scripting, web application, HTML, vulnerability

I. INTRODUCTION

Over the past few years, the development of information technology has been steadily rising, and a new level of detail in the structure has been reached. With rapid development, and the use of websites and their applications have become more widespread. It has both pros and cons. The list of pros is endless however, it is also of concern for the safety of the web. Due to transparency, ease of use, and development the safety concerns are becoming more challenging.[1]

According to SiteLock data, websites experience 22 attacks per day on average which are over 8,000 attacks per year. A website vulnerability is a flaw or misconfiguration in a web application code that allows an attacker to achieve a certain amount of control of the site, and possibly the hosting server. Many vulnerabilities are abused through automated means, such as vulnerability scanners and botnets. Cybercriminals develop specific tools that search the web for several platforms, like WordPress, exploring widespread and publicized vulnerabilities.[2] Once these vulnerabilities are found, they are then exploited to steal data, disperse malicious information, or inject defacement and spam content into the vulnerable site.

There are five common types of website vulnerabilities. These are frequently exploited. While the list is actually boundless, these five are the most common ones mentioned in Table 1.

S.no.	Vulnerability	Description
1.	SQL Injection (SQLi)	SQL injection vulnerabilities refer to fields in web application code where direct client input is forwarded to a database.
2.	Cross-Site Scripting (XSS)	Cross-site scripting befalls when attackers inject code scripts through common user input or other fields on a website to execute code. Cross-site scripting is utilized to concentrate on website users instead of the website or server itself.
3.	Command Injection	Command injection vulnerabilities authorize attackers to remotely pass and execute code on the hosting server of the website.
4.	File Inclusion	The include functions in server-side web application languages like PHP is used in Remote file inclusion (RFI) to execute code from a remotely stored file.
5.	Cross-Site Request Forgery	Cross-site request forgery attacks are less ordinary but can be quite harmful. CSRF attacks traps site users or administrators to inadvertently carry out malicious measures for the attacker.

Table 1. List of five common vulnerabilities

©IJRASET: All Rights are Reserved

In this exposition, the main focus is on Cross-Side Scripting (XSS) vulnerability. Cross-site scripting is a web security vulnerability. It allows an attacker to compromise the interactions that users have with a vulnerable application and bypass the same origin strategy. In cross-site scripting vulnerabilities, an attacker masquerades as a victim user and carries out any actions that the user can perform and accesses any of the user's data. If the victim user has gained access within the application, then the attacker might be able to get maximum control over all of the application's functionality and information.

In this paper, the implementation of a generic vulnerability scanner is present, along with its execution to find out XSS vulnerability in web applications. The remainder of the exposition is as follows: Section 2 contains the related work and section 3 comprises the proposed methodology. In section 4, the result and analysis of the keylogger are discussed. Section 5 concludes the paper with future scope. References are mentioned in section 6.

II. RELATED WORK

The concern for security in the network is increasing. Several approaches are offered out to protect the network from unauthorized access. Innovative methods have been implemented to find the potential inconsistencies that may harm the network. The vulnerability assessment is the most commonly used approach.[3][4].

The automated tools that define, detect, and categorize security flaws (vulnerabilities) in a computer, server, network, or communications infrastructure are known as a vulnerability scanner. Scanners detect missed patches on target machines and report linked and related vulnerabilities. Numerous current information security systems use vulnerability scanners as the central part of the risk assessment procedure. Others hinge on the output of the scanner in the systems patch management. [5] Web applications are becoming extremely prevalent in all kinds of business models and corporations. Currently, most vital systems such as those related to health care, banking, or even emergency response, are relying on these applications. They must therefore include, in addition to the anticipated value posed to their users, consistent and trustworthy mechanisms to safeguard their security.[6]

XSS remains a massive task for web applications, despite the many resolutions offered so far. There is no single solution that can efficiently diminish XSS attacks. More research is needed in the area of vulnerability elimination and depletion from the source code of the applications before implementation. [7]

III. PROPOSED METHODOLOGY

In this paper, the proposed vulnerability scanner is written in Python programming language. The software is generic but can be customized according to the vulnerability. This program takes the URL of a target website and maps the whole website. It then scans the website for weaknesses or vulnerabilities. Once done, it reports all the vulnerabilities it was able to find. This is done through HTTP requests.[8] In this, the user first clicks on the link, and the website generates a request. This request is then sent to the server where the server performs the request and sends the response back. In a cross-site scripting vulnerability, an attacker injects the code into the page using either forms or parameters in the link. The code is executed when the page loads. This is the fundamental functioning of the proposed vulnerability scanner.

A scanner class is created which is responsible for initiating target URL, target links, and links which are to be ignored. The class consists of various methods. The first method is used for extracting links from the form.

```
def extract_links_from(self,url):
        response = self.session.get(url)
        return re.findall('(?:href=")(.*?)"',response.content)
```

The second method is initiating the crawler function which is used to perform a deep scan of the website.

```
def crawl(self,url=None):
if url == None:
            url = self.target_url
        href_links = self.extract_links_from(url)

        for link in href_links:
            link = urlparse.urljoin(url,link)

            if "#" in link:
                link = link.split("#")[0]
```

©IJRASET: All Rights are Reserved

```
            if self.target_url in link and link not in self.target_links and link not in self.links_to_ignore:
                self.target_links.append(link)

                self.crawl(link)
```

The next two methods are used to extract and submit forms of the website.

```
def extract_forms(self,url):
        response = self.session.get(url)
        parsed_html = BeautifulSoup(response.content)
        return parsed_html.findAll("form")

    def submit_form(self,form,value,url):
        action = form.get("action")
        post_url = urlparse.urljoin(url,action)
        method = form.get("method")

        inputs_list = form.findAll("input")
        post_data = {}
```

The generic vulnerability scanner is customised to find XSS vulnerability using these two methods. One is used to find the vulnerability in a link and the other in a form.

```
def test_xss_in_link(self,url):
        xss_test_script = "<sCript>alert('test')</scriPt>"
        url = url.replace("=","="+ xss_test_script)
        response = self.session.get(url)
        return xss_test_script in response.content

    def test_xss_in_form(self,form,url):
        xss_test_script = "<sCript>alert('test')</scriPt>"
        response = self.submit_form(form,xss_test_script,url)
        return xss_test_script in response.content
```

Parameter	Value
Processor	Ryzen 5
RAM	8 gb
Operating system	Linux
Simulation Language	Python
Python version	3.8.6

Table 2. Simulation Parameters

IV. RESULT AND ANALYSIS

With the rapid development of the internet, web applications are becoming more integral. Therefore, more prone to attacks and vulnerabilities. In this exposition, a generic vulnerability scanner is customized to find cross-site scripting vulnerability in forms and parameters in a website. The scanner is also able to bypass the login page of a website and ignore information that is not useful in mapping a website. Fig 1. shows the mapped version of the website in which all the related links are listed. Followed by this in fig 2. in which two cross-site scripting vulnerabilities are listed. Fig 3. Illustrates the testing of other links which do not contain the cross-site scripting vulnerability.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

```
root@kali:~/PycharmProjects/vulnerability_scanner# python vulnerability_scanner.py
http://10.0.2.20/dvwa/dvwa/css/main.css
http://10.0.2.20/dvwa/favicon.ico
http://10.0.2.20/dvwa/
http://10.0.2.20/dvwa/instructions.php
http://10.0.2.20/dvwa/setup.php
http://10.0.2.20/dvwa/vulnerabilities/brute/
http://10.0.2.20/dvwa/vulnerabilities/exec/
http://10.0.2.20/dvwa/vulnerabilities/csrf/
http://10.0.2.20/dvwa/vulnerabilities/fi/?page=include.php
http://10.0.2.20/dvwa/vulnerabilities/sqli/
http://10.0.2.20/dvwa/vulnerabilities/sqli_blind/
http://10.0.2.20/dvwa/vulnerabilities/upload/
http://10.0.2.20/dvwa/vulnerabilities/xss_r/
http://10.0.2.20/dvwa/vulnerabilities/xss_s/
http://10.0.2.20/dvwa/security.php
http://10.0.2.20/dvwa/phpinfo.php
http://10.0.2.20/dvwa/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://10.0.2.20/dvwa/about.php
http://10.0.2.20/dvwa/instructions.php?doc=PHPIDS-license
http://10.0.2.20/dvwa/instructions.php?doc=readme
http://10.0.2.20/dvwa/instructions.php?doc=changelog
http://10.0.2.20/dvwa/instructions.php?doc=copying
http://10.0.2.20/dvwa/security.php?phpids=on
http://10.0.2.20/dvwa/security.php?phpids=off
http://10.0.2.20/dvwa/security.php?test=%22><script>eval(window.name)</script>
http://10.0.2.20/dvwa/ids_log.php
```

Figure 1. List of links

```
[***] XSS discovered in http://10.0.2.20/dvwa/vulnerabilities/xss_r/ in the follwing form
<form name="XSS" action="#" method="GET">
<p>What's your name?</p>
<input type="text" name="name" />
<input type="submit" value="Submit" />
</form>
[+] Testing form in http://10.0.2.20/dvwa/vulnerabilities/xss_s/


[***] XSS discovered in http://10.0.2.20/dvwa/vulnerabilities/xss_s/ in the follwing form
<form method="post" name="guestform" onsubmit="return validate_form(this)">
<table width="550" border="0" cellpadding="2" cellspacing="1">
<tr>
<td width="100">Name *</td> <td>
<input name="txtName" type="text" size="30" maxlength="10" /></td>
</tr>
<tr>
<td width="100">Message *</td> <td>
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></td>
</tr>
<tr>
<td width="100"> </td>
<td>
<input name="btnSign" type="submit" value="Sign Guestbook" onclick="return checkForm();" /></td>
</tr>
</table>
</form>
```

Figure 2. XSS vulnerabilities

```
[+] Testing form in http://10.0.2.20/dvwa/security.php

[+] Testing http://10.0.2.20/dvwa/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

[+] Testing http://10.0.2.20/dvwa/instructions.php?doc=PHPIDS-license

[+] Testing http://10.0.2.20/dvwa/instructions.php?doc=readme

[+] Testing http://10.0.2.20/dvwa/instructions.php?doc=changelog

[+] Testing http://10.0.2.20/dvwa/instructions.php?doc=copying
[+] Testing form in http://10.0.2.20/dvwa/security.php?phpids=on


[+] Testing http://10.0.2.20/dvwa/security.php?phpids=on
[+] Testing form in http://10.0.2.20/dvwa/security.php?phpids=off
```

Figure 3. Testing other links to find vulnerability.

©IJRASET: All Rights are Reserved

V. CONCLUSIONS

This paper studies the working principle of a generic vulnerability scanner along with its implementation to map out a website and find cross-site scripting vulnerability in it. Since the vulnerability scanner is a generic one it can be customized to find many vulnerabilities separately or together. By adding scalability, it can target multiple websites at the same time. Further, the code can be optimized to give a more efficient and effective mapping of the website. Moreover, the technology is not thorough enough and can be further enhanced. Research on XSS is still dynamic with publications across numerous conference proceedings and journals. The areas focused on by most of the studies are attack prevention and vulnerability detection. Dynamic analysis techniques form the bulk of the resolutions offered by a variety of studies.

REFERENCES

[1] B. Wang, L. Liu, F. Li, J. Zhang, T. Chen and Z. Zou, "Research on Web Application Security Vulnerability Scanning Technology," *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2019, pp. 1524-1528, doi: 10.1109/IAEAC47372.2019.8997964.

[2] W. Qianqian and L. Xiangjun, "Research and design on Web application vulnerability scanning service," *2014 IEEE 5th International Conference on Software Engineering and Service Science*, 2014, pp. 671-674, doi: 10.1109/ICSESS.2014.6933657.

[3] Bairwa, Sheetal & Mewara, Bhawna & Gajrani, Jyoti. (2014). Vulnerability Scanners-A Proactive Approach To Assess Web Application Security. International Journal on Computational Science & Applications. 4. 10.5121/ijcsa.2014.4111.

[4] Rubin A, DanielGeer, Ranum M. Web Security Sourcebook[J]. Edpacs, 1997, 25(8):14-15

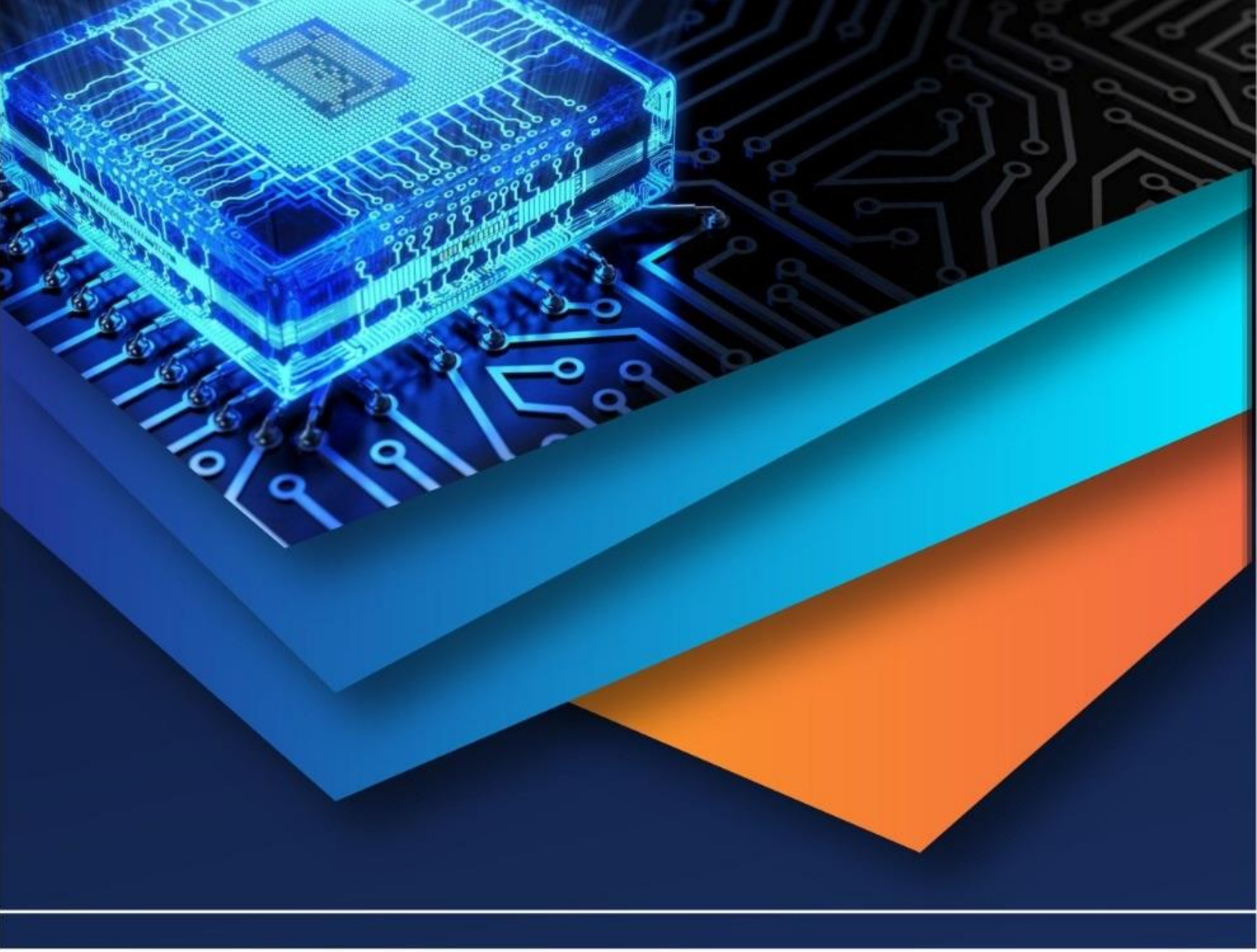
[5] Badawy M.A., El-Fishawy N., Elshakankiry O. (2013) Vulnerability Scanners Capabilities for Detecting Windows Missed Patches: Comparative Study. In: Awad A.I., Hassanien A.E., Baba K. (eds) Advances in Security of Information and Communication Networks. SecNet 2013. Communications in Computer and Information Science, vol 381. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40597-6_16

[6] Garcia-Alfaro, Joaquin & Navarro-Arribas, Guillermo. (2009). A Survey on Cross-Site Scripting Attacks.

[7] Isatou Hydara, Abu Bakar Md. Sultan, Hazura Zulzalil, Novia Admodisastro, Current state of research on cross-site scripting (XSS) – A systematic literature review, Information and Software Technology, Volume 58, 2015, Pages 170-186, ISSN 0950-5849,

[8] Ford, J.D., Pearce, T., McDowell, G. *et al.* Vulnerability and its discontents: the past, present, and future of climate change vulnerability research. *Climatic Change* **151,** 189–203 (2018). https://doi.org/10.1007/s10584-018-2304-1

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)