International Journal of Safety and Security Engineering
Vol. 14, No. 2, April, 2024, pp. 523-532
Journal homepage: http://iieta.org/journals/ijsse

An Experimental Study on Detecting and Mitigating Vulnerabilities in Web Applications

Rishith Pranav Kumar Kollepalli[1], Mallidi Jaswanth Srinivasa Reddy[1], Bellam Lakshman Sai[1], Abirami Natarajan[1], Senthilkumar Mathi[1*], Venkadeshan Ramalingam[2]

[1] Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore 641112, India
[2] Faculty in Information Technology Department, University of Technology and Applied Sciences-Shinas, Shinas 324, Oman

Corresponding Author Email: m_senthil@cb.amrita.edu

https://doi.org/10.18280/ijsse.140219

Received: 5 September 2023
Revised: 21 March 2024
Accepted: 28 March 2024
Available online: 26 April 2024

Keywords:
ethical hacking, penetration testing, security, vulnerability, web application

ABSTRACT

The increasing use of the internet has led to a growing number of security threats. Computers, smartphones, smartwatches, and other mobile devices associated with the internet face different threats and exploits. In those cases, different services are provided through web applications only. Those applications are vulnerable to hacking. There are over 1.9 billion websites today, and everything is connected to the network. According to the new national vulnerability database update, 10,683 weaknesses were found in web applications in the first quarter of 2023. The websites have the most significant details of the clients, like personal details, financial details, and so on. Checking all the web application weaknesses is not a silver bullet. So, vulnerability scanners play a significant role in web application security. Vulnerability analysis and penetration testing are two distinct vulnerability types of testing. These tests can help identify all the vulnerabilities in a web application, even those not detected by vulnerability scanners. While certain users access this vulnerability analysis data with just honest goals, like creating some security measures to avoid those vulnerabilities, some utilize it to recognize ways of destroying significant information and records of websites. As it is notable, the term penetration testing is also ethical hacking. The current paper aims to investigate penetration testing on web applications. The paper discusses the different types of penetration testing, the tools and techniques used, and the benefits of penetration testing. It also suggests the challenges of penetration testing and the steps that can be taken to mitigate these challenges.

1. INTRODUCTION

Information security is essential for all firms due to the rise in cyber hacking activities in the modern digital age [1]. Irrespective of their size, every business is faced with the imperative task of safeguarding their data against potential attackers. Web applications play a major role in communication, resource sharing, social networking, online banking, e-commerce, etc. [2]. Cybercriminals target web application vulnerabilities to gain unauthorized access to sensitive data, compromising user privacy and security. A company that develops web applications covers most of these features. So, these are the essential details for the attacker to start their attack on that company. Because of that, the attackers mostly target web applications. If the attacker hacked the web application, they could access all the confidential or sensitive data available in the web application. It leads the attack to the next stage of an attack [3]. For example, if an online shopping website's credit or debit card details are saved, and the attackers hack the website, they can obtain the full card details and use them to steal money from the victim's account.

Maintaining good security mechanisms to avoid cyber threats is challenging [4]. The first step in achieving this is to identify any vulnerabilities or security loopholes in the system. These can be identified with the help of some testing techniques. Distinguishing between Vulnerability Assessment and Penetration Testing reveals two distinct approaches to testing methodologies. Vulnerability assessment is a technique to identify weaknesses or measure security vulnerabilities. Penetration testing takes it to the next level, checking everything – software, hardware, the whole setup to find out where things could go wrong [5]. It is basically like trying to think like a cyber attacker about what they could get into if they broke through the system's defenses [6]. Penetration testing for web applications has been an active area of research in recent years, with researchers proposing new methodologies, tools, and case studies. Some studies have focused on specific types of web app vulnerabilities like SQL injection [7] and XSS [8], while others have taken a more comprehensive approach covering various vulnerabilities [9-12]. The use of automated scanning tools like Nmap, Nikto, and ZAP, along with manual verification, has been commonly employed. However, the dynamic nature of web applications and the continuous emergence of new vulnerabilities necessitates constant research efforts.

Performing manual penetration testing is daunting as the tester must conduct every possible test to find a single vulnerability. In one application, many vulnerabilities may

exist. It is not an easy task, so the penetration tester uses automated tools to avoid that. In automated penetration testing, few tools are available as open-source, and some as commercial products with different functionalities [13]. The main problem in this automated penetration testing is in selecting the best tool. The tester can't say that this tool finds all the vulnerabilities in the web application. The vulnerability findings differ for each tool. None of the tools can identify the complete vulnerability in the web application.

The current research aims to explore the vulnerabilities present in web applications, their impact on security, and best practices to mitigate risks. The investigation aims to conduct a comprehensive penetration testing of an online shopping web application, evaluate its security posture, and provide recommendations to enhance its security defenses.

The literature survey is discussed in section 2, methodologies are discussed in section 3, prevention techniques are briefly explained in section 4, and finally, the result is concluded in section 5.

2. LITERATURE SURVEY

In the work of Aibekova and Selvarajah [9], the author discussed different penetration testing attacks and the types in detail using Kali Linux. The process was done in six phases: Reconnaissance, Enumeration, Exploitation, Dictionary attack, Privilege Escalation, and DOS Attack. In the reconnaissance phase, they identified the open port details by scanning the target IP address in the Nmap tool and the next phase, enumeration, and they started tracking the IP address details like subnets, hosts, interfaces, DNS records, web server details, etc., using OpenNetAdmin tool. In the second phase, they found details about the web server. In the exploitation phase, they used the Metasploit tool to identify exploits like buffer overflow, code injection, and web application. In the fourth phase, they started using a dictionary attack, in which they identified a hashed password file. With that hashed password, they used John the Ripper to decrypt it and got the target's exact password. Now, they moved on to privilege escalation. In this phase, they explored each user; after collecting all the necessary resources, they started a DoS attack against the target.

The security evaluation was conducted of a web application that employs penetration testing strategies to showcase man-in-the-middle attacks [10]. They used a few penetration testing strategies like SQL Injection, cross-site scripting (XSS), and brute force attacks. They started the evaluation with information gathering using the Nmap tool and got information like port, state, service, and version. Using that Nmap output, vulnerability analysis was initiated using Nikto, W3af, and Zaproxy tools. Nikto finds 8 loopholes in the target, Zaproxy finds 7 loopholes in the target, and W3af fails to find the loopholes. During exploitation, the team identified an SQL injection error, XSS exploit, password dictionary, brute force attack process on burp suite, and a sniffing attack with Wireshark. After the automatic penetration testing was completed, manual penetration testing was performed, and the findings were compared to the percentage range.

In the investigation of Ojagbule et al. [7], the author compared SQL vulnerabilities found in common content management systems. To achieve this, the Nikto tool was used for the initial scanning phase, and the SQLMAP tool was used for penetration testing. The three content management systems analyzed were WordPress, Joomla, and Drupal, all installed on localhost. There are four phases involved in penetration testing in this paper. The first phase is gathering information about the system and server to be targeted, then proceeding with the scanning phase; they used a Nikto scanner to check dangerous files, outdated versions, and server problems. After scanning, they started with exploitation using SQLMAP, and the final phase is mitigation, which comprises ways of removing the vulnerabilities found during the previous phases and the prevention techniques. In the whole process, the scanning didn't show any vulnerable SQL attacks; instead, it warned about the other vulnerabilities available on those websites, WordPress, Joomla, and Drupal.

The investigation focused on the client and server-side attack possibilities in Kali Linux [11]. The main advantage of using Kali Linux is its many built-in hacking tools, which significantly help us perform vulnerability analysis and security testing. In the context of client-side attacks, the authors discuss the need for such attacks when server-side attacks fail or when the attacker fails to obtain the proper IP address. The attackers use social engineering techniques to get important user details like names and social media account details. From those details, they start analyzing the user. It is helpful for them to understand more about the client. The most common client-side attack they mentioned is the insertion of Trojans into the device, and they even mentioned some prevention techniques for the attack. They discussed Netdiscover and Zenmap's tools in server-side attacks because they don't require many details. Only the target IP is enough. Those tools can use the IP address to identify details like open ports, OS details, and installed services and works. After all this discussion, they demonstrated packet sniffing, DoS attacks, man-in-the-middle attacks, and fake access points.

Devi and Kumar [12] proposed identifying web application security weaknesses using ethical hacking techniques. For this assessment, they used tools like Nikto, OWASP's Zed attack proxy, Netcraft, Sparta, and Nmap in the Kali Linux platform. They detected security weaknesses in all domains from low to medium with OWASP's ZAP tool. The detected vulnerability is mostly cross-site scripting, SSL, server leak, HTTP header, Retrieved x-powered, cookie without secure flag, URL rewriting, application error disclosure, etc. As a result, they concluded that Nikto found more vulnerabilities than OWASP's ZAP tool.

In the work of Anand and Singh [13], penetration testing was done for Amazon Echo against a denial-of-service attack on the Kali Linux platform. They started the testing by assuming that the cyber criminals already have access to the home network. They monitored the whole network traffic of the Amazon Echo device under DoS attack, one case to perform the attack on Kali Linux and one more instance for monitoring the network during the attack on the same Kali Linux. These attacks cause the device to crash, disconnecting the device from the network. This process is monitored using the Wireshark tool in Kali Linux, and they also showed a network packet drop during the attack. By doing all these processes, they found that initializing the denial-of-service attack in Amazon Echos is easier. The only thing that attackers should know is to access the home network, which leads them to get the required information about the devices connected to the network. They gathered all the information about the device using Nmap for their demonstration. After getting the device's MAC address, they used the SPARTA tool to identify the open ports. The airodump is used to obtain basic service

set identification of the router. Finally, they started attacking the device using the Metasploit tool in Kali Linux.

The main aim of the previous study is to develop a fake website using Python scripting and a flask server to detect the SQL injection, cross-site scripting, spider content discovery tool and skip fish in Kali Linux [14]. They used a spider to discover the content and functionality of the visible content where the users can browse. XSS and SQL injections are detected in the forms. The authors also conducted automated scans using technologies such as skipfish to give an in-depth website analysis. The main advantage of using skip fish is after completing the scan, it creates a separate zip file for the result. In that file, "index.html" may exist. It contains all the vulnerability details that it found during the scan. After doing all these automated scans, they tried manual testing like SQL injection, cross-site scripting, stored XSS, and Command Injection. As a prevention technique, they requested parameterized queries in SQL so that the attacker cannot manipulate the query as they wish in SQL injection. Additionally, using Content Security Policies (CSPs) in the response header is recommended to prevent cross-site scripting. This paper discusses the most frequent vulnerabilities and automated preventative methods.

In the work of Kandasamy et al. [15], the author investigated the recent cyber-attacks on healthcare institutions in Asia. Their investigation found five different types of attacks, mainly dominating healthcare institutions. This paper discusses those five attacks, their vulnerabilities, and their risks. The five main types of cyber-attacks are the Trojan attack, Phishing attack, Ransomware, Advanced Persistent Threat (APT), and Malware - Credentials compromise. There are four main types of Malware: Malicious software is intended to get unauthorized access to someone's system without their knowledge. The three types of Malware are Trojan, Spyware, and Ransomware. In 2021, ransomware attacks targeting healthcare institutions witnessed a significant surge of 150%, as reported by the Cybersecurity and Infrastructure Security Agency (CISA). As per the report's findings, healthcare institutions faced an average cost of $1.1 million due to ransomware attacks.

Phishing attacks are also social engineering attacks used to steal user details, including financial details like credit card numbers, by sending malicious links to the users. A recent study shows Malaysia ranks third in Asia for phishing attacks. Advanced Persistent Threat is deployed over a long period. It is a selective attack to obtain unauthorized access to the organization's confidential data. APT has eight stages: Initial Recon, Initial Compromise, Establish Foothold, Escalate Privilege, Internal Recon, Move Laterally, Maintain Presence, and Complete Mission. During this COVID-19 period, this was the most common attack in Asian Healthcare institutions by some Chinese APT groups, and they got around 68 lakhs of patient details. After discussing all the attacks, they listed the most common vulnerabilities that cause them with the help of the national vulnerability database and common vulnerabilities and exposes.

The motive in the study of Gunawan et al. [16] is to penetrate web servers by simulating SQL injection, Cross-site scripting, and WordPress attacks. With the help of Burp Suite and SQLMap, they exploited the SQL injection. Penetration testing is used to scan and gather information from web applications for attacks. It gathers HTTP GET and POST requests from the web server and the website cookies, essential to accessing the SQL database. For XSS, they used the BeEF tool on Kali Linux. They made a small modification to the technique used in this attack. They created a phishing website that redirects the site to another phishing page, which helps the attacker download the required software and make it on the victim's machine.

In the investigation of Kumar and Tlhagadikgora [17], the author implemented network and system administration penetration testing. They designed and set up a virtual network laboratory to conduct penetration testing by demonstrating the attacks using Kali Linux. Four phases are involved in this paper: information gathering, vulnerability analysis, exploitation, and reporting. The initial phase involved information gathering, utilizing three open-source tools available in Kali Linux: i.e. Nmap, Zenmap, and Dmitry. Nmap and Zenmap identify the live host in the virtual network laboratory. Once the live hosts were identified, the next step involved scanning the open ports using OS and service fingerprinting techniques. After that, they started the vulnerability analysis using Nexpose community, Nessus, and OpenVAS. The report generated by Nessus contains a list of vulnerabilities against every host.

Similarly, OpenVAS worked under the same configuration, allowing for a comprehensive assessment of the security posture of the organization's IT infrastructure. By comparing the scanning reports generated by these tools, the author identified various vulnerabilities, including buffer overflow, spoofing, denial-of-service (DoS), and privilege elevation. It helps the organization find loopholes exploited by the attacker so they can develop a security mechanism for the loopholes.

3. METHODOLOGY

3.1 Research design

The present work follows an experimental design approach to assess the vulnerabilities and perform penetration testing on a dummy shopping website. It evaluates security measures put in place methodically by manipulating controlled variables. Identifying potential vulnerabilities in the target web application was mainly done with the Nikto vulnerability scanner during the vulnerability detection phase. It is an automated vulnerability evaluation that scans for recognized vulnerabilities, misconfigurations, and security weaknesses without exploiting them. The results from Nikto showed issues such as the absence of security headers, old software versions, and configuration problems that may be vulnerable to exploitation.

While vulnerability detection finds possible weaknesses, penetration testing goes a step beyond by actively trying to exploit those vulnerabilities. In this research, various penetration testing techniques were employed:

1) Phishing attacks using SetoolKit to test for broken authentication and sensitive data exposure vulnerabilities.
2) Using BurpSuite to modify and tamper with server requests, simulating attacks like price manipulation or unauthorized actions.
3) Server-Side Request Forgery attacks to test for the ability to access restricted resources and sensitive data.

These hands-on exploitation attempts went beyond detecting vulnerabilities and validated their exploitability and impact in a controlled environment.

3.2 Target environment

The target of this study is a dummy shopping website created specifically for this purpose. A Whois lookup revealed that the website is hosted on a server with IP address 65.8.178.77.

3.3 Information gathering

The present paper uses Nmap to gather information about the target website. The few important features of Nmap are host discovery, Port Scan, Service and Version Detection, and Operating System detection [18]. Before proceeding with Nmap commands, Whois lookup is used to identify the IP address of the website.

4 ping borerases
177 (85.8.178.77) 56(84) bytes of data.

— 65.8.178.77 ping statistics —
48 packets transmitted, 0 received, 100% packet loss, time 4sssens

(kali® kali)-(~]
nmap -A 65.8.178.77
Starting Neap 7.92 (https://nmap.org) at 2022-07-03 08:29 EOT
Naap scan report for server-63-8-176-7).mia3.r.cloudfront net (65.8.178.77)
Host is up (0.9545 latency).
Not shown: 982 closed tcp ports (conn-refused)

PORT	STATE	SERVICE	VERSION
22/tcp	Filtered	ssh	
80/tcp	open	http	Amazon CloudFront httpd

_http-title: ERROR : The request could not be satisfied
_http-server-header: CloudFront

443/tcp	open	ssl/https?
445/tcp	filtered	microsoft-ds
1718/tep	filtered	h323gatedisc
1719/tep	filtered	h323gatestat
5222/tcp	filtered	xmpp- client
8000/tcp	filtered	http-alt
8001/tcp	filtered	veon- tunnel
8022/tcp	filtered	oa- system
8045/tcp	filtered	unknown
8080/tcp	filtered	http- proxy
8081/tcp	filtered	blackice-icecap
8100/tcp	filtered	xprint-server
8888/tcp	filtered	sun-answerbook
10001/tcp	filtered	scp-config
10002/tcp	filtered	documentum
10003/tcp	filtered	documentum_s

Service detection performed. Please report any incorrect results at https://neap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 27.51 seconds

Figure 1. Service and version detection

Prior to executing Nmap commands, a Whois lookup is performed to identify the IP address of the target website. Subsequently, Nmap is utilized for the following tasks using the obtained IP address:

1. Nmap leverages host discovery algorithms to detect active hosts within a specified IP range. This step is crucial in identifying live systems for further probing.
2. Port Scanning is conducted to determine which ports on the target system are open and potentially hosting services accepting connections. Nmap's port scanning capabilities help uncover potential attack vectors.
3. For the open ports identified during port scanning, Nmap attempts to identify the specific services running on those ports and their respective versions. Accurate service and version information is valuable for assessing potential vulnerabilities associated with outdated or insecure software.
4. Nmap incorporates techniques for detecting the operating system running on the target system. If direct identification is not possible, Nmap provides educated guesses based on its fingerprinting database, along with a confidence rating for each guess. Knowledge of the underlying operating system aids in understanding potential vulnerabilities and tailoring further testing activities.

The information gathered through these Nmap features serves as a foundation for subsequent phases of the vulnerability assessment and penetration testing process.

[sudo] password for kali:
(root@kali)-[~]
Nmap –0 65.8.178.77
Starting Nmap 7.92 (https://nmap.org) at 2022-07-03) 08:59 EDT
Nmap scan report for server-65-8-178-77.mia3.r.cloudfront.net (65.8.178.77)
Host is up (0.0047s latency).
Not shown: 982 closed tcp ports (reset)

PORT	STATE	SERVICE
22/tcp	filtered	ssh
80/tcp	open	http
443/tcp	open	https
445/tcp	filtered	Microsoft-ds
1718/tcp	filtered	h323gatedisc
1719/tcp	filtered	h323gatestat
5222/tcp	filtered	Xmpp-client
8000/tcp	filtered	Http-alt
8001/tcp	filtered	Vcom-tunnel
8022/tcp	filtered	Oa-system
8045/tcp	filtered	unknown
8080/tcp	filtered	Http-proxy
8081/tcp	filtered	Blackice-icecap
8100/tcp	filtered	Xprint-server
8888/tcp	filtered	Sun-answerbook
10001/tcp	filtered	Scp-config
10002/tcp	filtered	Documentum
10003/tcp	filtered	Documentum_s

Device type: VoIP phone|webcam|specilized|firewall|general purpose
Running(JUST GUESSING): Grandstream embedded (92%), Garmin embedded (89%) ,2N embedded (88%), Firebrick embedded (86%) ,NodeMCU embedded (85%) , lwip 1.4.X (85%), cognex embedded (85%)
OS CPE: cpe:/h:grandstream:gxp1105 cpe:/h:garmin:virb_elite cpe:/h:2n:helios cpe:/h:firebrick:fb2700 cpe:/o:nodemcu:nodemcu cpe:/a:lwip_project:lwip cpe:/h:philips cpe:/a:lwip_project:lwip:1.4
Aggressive OS guesses: Grandstream GXP1105 VoIP phone (92%), Garmin Virb Elite action camera (89%), 2N Helios IP VoIP doorbell (88%), FireBrick FB2700 firewall(86%), firmware (lwiP stack) (85%). Philips Hue Bridge (lwip stack v1.4,,o) (85%), Cognex Dataman 200 ID reader (lwiP TCP/IP stack) (85%)

OS detection performed. Please report any incorrect results at https://nmap.org/submit/.
Namp done: 1 IP address (1 host up) scanned in 8.47 seconds

Figure 2. OS detection

Figure 1 shows the output of service and version detection in Nmap. Of 1000 ports, 982 are closed, so it is not showing anything about them. The remaining ports show the details; from the output, port 80/TCP, the HTTP service, is in an open state with Amazon CloudFront httpd. Figure 2 shows the operating system detection output. The Nmap fails to detect the correct operating system for the target website. Instead, it lists the possible operating system based on the guess. According to the guess, the operating system was Grandstream

embedded.

3.4 Vulnerability scanning

Several activities include crawling, fuzzing, and analyzing web application vulnerability scanning. Crawling searches for pages in a web application, fuzzing simulates the attack. Finally, the analyzer analyses the response given by the fuzzer to identify whether the information is vulnerable. Three methodologies are used in vulnerability finding for the web application, namely, Black-Box Testing when the scanner doesn't have any information about the web application, White-Box Testing when the scanner knows some information about the web application, and grey-box testing to check the output based on the given input with some limited knowledge of web application [19]. This paper uses the Nikto vulnerability scanner to identify the vulnerability in the web application.

Nikto is an open-source vulnerability scanning tool available in Kali Linux, which can test more than 6700 harmful files or programs and more than 1200 outdated version detection [20]. Figure 3 shows the Nikto vulnerability scanner output. It found seven pieces of information about the target web application. The first information is anti-clickjacking X-Frame-Options header is not present, which means the attacker can use this web application to trick a user by clicking a button or a link that redirects the user to a malicious website. The next vulnerability in that web application is the X-XSS protection header. This header configures the reflective Cross-Site Scripting protection; it helps us stop page loading if it finds XSS attacks. This protection header is not defined in this web application, so the web page loads if the attacker gives any Cross-Site Scripting into the web application. Because of that, the attacker can achieve their goal. The content-type response header is not set in the webpage, and this response header is responsible for protecting Multipurpose Internet Mail Extensions (MIME) sniffing vulnerabilities. MIME sniffing vulnerabilities occur when a web application allows users to upload content. The attacker can upload the malicious file if it allows the user to upload content. Along with these vulnerabilities, a few more uncommon headers are also identified.

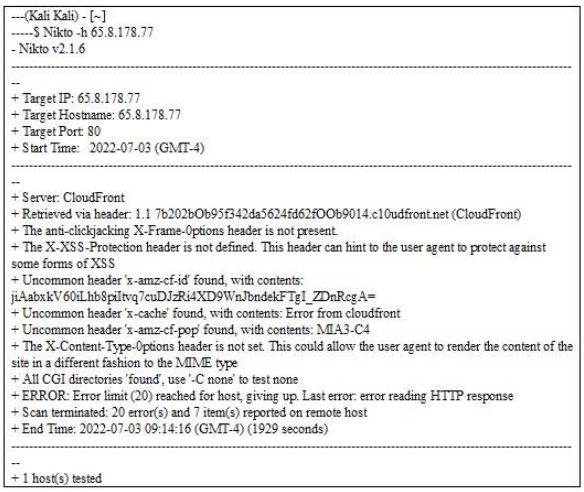

Figure 3. Nikto vulnerability analysis

The "X-Amz-Cf-Id" header is unique to Amazon Web Services (AWS) CloudFront, a content delivery network (CDN) service offered by Amazon and is not a standard HTTP header. CloudFront produces this header and is present in the server's HTTP response. The "X-Amz-Cf-Id" header in the output of the Nikto vulnerability scanner does not signify a vulnerability. As was already noted, the content delivery network Amazon Web Services (AWS) CloudFront produces the "X-Amz-Cf-Id" header, which is a unique header. Another distinctive header associated with Amazon Web Services (AWS) CloudFront is the "X-Amz-Cf-Pop" header. The "CloudFront Point of Presence" (POP) header is abbreviated as such. Figure 4 shows the w3af vulnerability scanning.

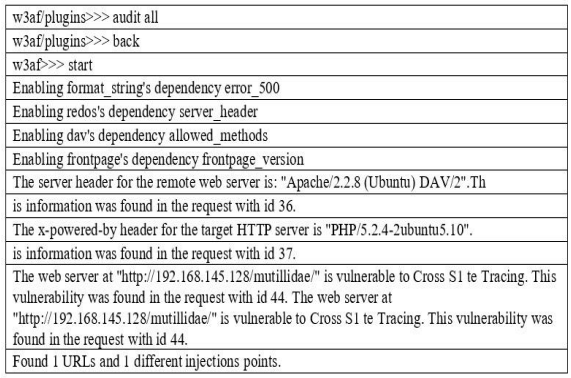

Figure 4. w3af vulnerability scanning

The key findings of using w3af are as follows.

Wildcard DNS Configuration: The target website is using a wildcard DNS configuration, where multiple subdomains resolve to the same IP address (192.168.153.1). This configuration can probably reveal sensitive statistics, and boom the assault floor.

Sensitive Information Disclosure: The experiment located numerous URLs related to the phpinfo.Personal home page script, which could display sensitive information approximately the server configuration, including set up modules, surroundings variables, and paths.

Potential XSS Vulnerabilities: The test identified potential Cross-Site Scripting (XSS) vulnerabilities that can allow an attacker to inject malicious scripts into the website and doubtlessly compromise consumer money owed or scouse borrow sensitive facts.

Potential XST Vulnerability: The scan detected a potential Cross-Site Tracing (XST) vulnerability, which could enable an attacker to trace the website's requests and potentially gain access to sensitive information.

Deliberate Vulnerable Application: The scan found a URL (http://192.168.145.128/mutillidae/) that is likely a deliberately vulnerable web application used for testing and learning purposes.

On a different website, information along with the DAV methods enabled on the target HTTP server and the scan results. The vulnerabilities scan using w3af is listed in Table 1.

Table 1. Vulnerabilities scan using w3af

Header Type	Value
Server Header	Microsoft-IIS/8.5
x-aspnet-version	4.0.30319
x-powered-by	ASP.NET

Target URL with DAV methods enabled:

- URL: http://www.acuart.com/
- DAV methods enabled, such as baseline control, check-in, checkout, connect, copy, debug, get, head, index, invalid, invoke, label, link, lock, merge, subscribe, subscriptions, text search, trace, track, unlink, unlock, unsubscribe, and version_control

Scan Results:

- Found 1 URL: http://www.acuart.com/
- Different injection points: 1

The scan enables various dependency modules or plugins related to error handling, server headers, allowed methods, and version information for different components like "format string", "redo's", "dav", and "frontpage".

The scan detected information about the target HTTP server and its version in the requests with specific IDs (36 and 37). The detected server is "Microsoft-IIS/8.5", and the "X-AspNet-Version" header shows the server version as "4.0.30319". Additionally, the "X-Powered-By" header indicates the target server is "ASP.NET".

For the URL "http://www.acuart.com/", the scan found a list of enabled DAV (Distributed Authoring and Versioning) methods. The enabled methods include various http methods like acl, baseline control, checkin, checkout, connect, copy, debug, get, head, index, invoke, label, link, lock, merge, mkactivity, mkcol, move, notify, options, patch, pin, poll, post, prop find, prop patch, put, report, rmdir, search, show method, spacejunc, subscribe, subscribe tree, text search, trace, track, uncheck out, unlink, unlock, unsubscribe, and version control. The scan found 1 URL ("http://www.acuart.com/") and 1 different injection point during the scan, which finished in 38 seconds.

3.5 Exploitation

Attackers possess various tactics to exploit vulnerabilities and compromise system security. These include manipulating SQL queries for data extraction, exploiting authentication weaknesses for unauthorized access, and intercepting sensitive information. Phishing attacks trick people into giving up login details or personal information, while server-side request forgery (SSRF) attacks provide unauthorized access to company property. Cross-site scripting (XSS) flaws enable session hijacking when DoS attacks destroy systems. File attachment vulnerabilities can lead to increased access or privilege, while broken access allows unauthorized access. XML External Entity (XXE) attacks manipulate XML input for code execution. These strategies highlight the diverse approaches attackers use to exploit system weaknesses and compromise security.

The most exploited vulnerabilities are SQL injections, Broken Authentication, Sensitive Data Exposure, File inclusion testing, Server-Side Request Forgery, XML external entities, broken access control, security misconfigurations, and cross-site scripting. The present paper is mainly focusing on broken authentication and sensitive data exposure.

One of the most common broken authentication attacks is a phishing attack. The phishing attack is one of the cybersecurity attacks used to steal user details, including login credentials and even credit card numbers. The attacker sends malicious messages as a trusted entity with malicious links for this attack. Those links lead the user to install Malware software or files that freeze the system as part of a ransomware attack, or the attackers can steal sensitive information [21]. Phishing for 2FA codes (two-factor authentication). Attackers may try to trick users into submitting their 2FA codes or completely circumvent the 2FA procedure when a website uses 2FA. This can be accomplished by pretending to be reputable services or by utilizing social engineering strategies to get consumers to divulge their codes. This paper uses the SetoolKit open-source tool for a phishing attack. This tool clones the same website created for this paper. If the user enters it, the attacker can get all the details about the user system and download the required malicious files or code into the target system. But as far as the target is concerned, they cannot differentiate between the original and phishing sites because they look the same. Phishing for personal information, attackers may ask consumers to disclose sensitive data through false emails or communications that appear to be sent from a reliable source. Examples are requests for private company data, personnel records, or customer databases.

The next tool is Burpsuite, one of the most popular web attack platforms, which can be used for scanning and even attacks [22]. Using this tool, the tester can simulate various security threats to test their application, which helps them improve their security policy. In the burp suite, the important feature is the agent function, which uses a proxy module between the web application and the server. The proxy server is a server between the web browser and the server. When the client uses the browser instead of using the correct server, it tries to retrieve the information from the proxy server. The proxy server retrieves the correct server's information and displays it to the client. Because of that, the attacker looks through the proxy server's information. Even the attacker can modify the request in the web application; in this attack, the attacker modifies the content in the browser like they are trying to reduce the product price or change the product in the cart. Like that, many vulnerabilities can be found using penetration testing on the web application.

In a server-side request forgery (SSRF) attack, resources are accessed or changed by the attacker employing server capability that has been abused. The attacker's target is A program that permits data imports from URLs or enables users to read data from URLs. Changing URLs or messing with URL path traversal to modify URLs is possible. Attackers often provide a URL (or alter an existing one), and the server's running code reads from or submits data to it. Attackers can use URLs to access private information and services, such as HTTP-enabled databases and server configuration data, not intended to be made public. An attacker can include a file via the File Inclusion vulnerability, usually by taking advantage of "dynamic file inclusion" procedures set up in the target programme. The usage of user input without enough validation leads to vulnerability. This may result in the file's contents being produced, but depending on how serious the problem is, it may also result in. execution of code on the web server. Cross-site scripting (XSS) attacks result from client-side code execution, such as JavaScript.DoS attacks and the disclosure of sensitive information.

4. PREVENTION TECHNIQUES

One of the vulnerabilities in web applications is cross-site scripting (XSS); as said before, this is a client-side code injection attack in which the attacker executes malicious code in the web application.

Input validation and sanitization of user input by removing or encoding potential malicious scripts like <script> tags is

crucial to prevent XSS. Ensure output encoding of any user-supplied data before rendering to prevent injected scripts from executing. Implement a Content Security Policy (CSP) to whitelist trusted content sources and turn off inline script execution, making it harder for malicious scripts to get injected. Configure browser security headers like X-XSS-Protection to enable XSS filtering, instructing browsers to block pages with code injection attack signatures.

Preventing the XSS attack is not easy, but if the developers follow a few steps, they can still prevent the XSS attack [23]. The first step is to properly train the developers about the XSS attack, then ask them to use QA staff, DevOps, and SysAdmins. The most important thing is the developers need to treat all user input as public input. It is impossible to identify how the attacker tries to access a web application, and the attacker can access the web application even with an authorized login, so treating all the users as public input is the main aspect. Setting the HttpOnly flag for cookies helps us avoid accessing client-side JavaScript. Scanning the web application also helps us identify the XSS attacks [24].

Some prevention techniques can be used for XSS scripting to avoid these attacks. However, preventing phishing attacks is more difficult. The only prevention the client can do is click any link from any source, even if the sender is known to the client. Instead of clicking the link, the user can use the search engine to access the site. Nowadays, most browsers enable add-ons to help us spot malicious websites or alert users about phishing sites [25]. Most importantly, don't give any personal or sensitive information on an insecure site, change the password regularly, and don't keep the same password for all websites. If the password were the same, it would be easy for the attacker to log in to any user site.

Compared with previous attacks, preventing SQL injection is simple to implement. The SQL injection attacks can be prevented using prepared statements with parameterized queries, properly constructed procedures, escaping all user-supplied input, and enforcing least privilege [26]. These are the few defence techniques that can prevent SQL injection attacks.

Input validation and sanitization of user input before database queries prevent SQL injection attacks. Use least privileged database accounts with only required permissions. Utilize stored procedures for database access as they are less vulnerable when coded properly. Implement multi-factor authentication beyond just passwords. Enforce strong password policies for length, complexity, and rotation. Automatically lock accounts after failed login attempts to prevent brute-forcing. Implement a centralized authentication system across applications instead of ad-hoc logic. These are the few defence techniques that can prevent SQL injection attacks.

One prevention technique is installing a firewall, which is an efficient way to prevent an attack; it works as a shield between the system and the attacker. Next, click on any advertisements or pop-ups that can be avoided in the web application because they may contain malicious code. The malicious code is automatically downloaded into the system without anyone's knowledge [27]. Never utilize the input directly in the application code. Not just online form inputs like login forms, but all input must be sanitized by the developer. They must eliminate components of potentially dangerous code, such as single quotations. It's a good idea to turn off the display of database problems on the live website. SQL injection can gather details about the database via database problems. All user-supplied input fields should undergo stringent validation, especially those requiring network requests to prevent server-side request forgery validation of input.

Technique for preventing server-side request forgery Validation of Input All user-supplied input fields should undergo stringent validation, especially those that require network requests. To make sure the URLs or IP addresses are authentic and authorized, validate and sanitize them. Assign the program access to a whitelist of permitted domains, IP addresses, and protocols. Check any user-provided URLs against this whitelist to limit requests to just reputable resources. Filtering based on a whitelist is an effective way to prevent SSRF attacks. Maintaining an Allowlist of permitted URLs/IPs your application can access is a key SSRF prevention measure. Reject any requests attempting to access resources outside of the pre-approved allowlist. Segregating the Application Logic that integrates remote resources loaded over the network can contain the scope of potential SSRF vulnerabilities and attacks within your environment.

Cross-Site Request Forgery (CSRF) tokens can be used to verify the integrity of requests coming from your application. As a result, attackers are less likely to deceive users into sending unauthorized requests that could result in SSRF assaults. Whitelisting of URLs on the server: implement server-side URL whitelisting if your application requires the ability to retrieve resources from external URLs. Keep a list of permitted URLs and only retrieve resources from those sources.

5. RESULTS ANALYSIS

In this section, three different vulnerability scanners, Nmap, Nessus, and Nikto, are compared regarding features and the vulnerabilities they can find. Each tool is compared with features like cost, usage, capable operating system, and advantages and disadvantages in a feature comparison.

The other comparison is purely based on the vulnerability it can find; not all vulnerability scanners are the same, and each scanner has some unique features.

Nmap Port Scanning Results:

- Out of 1000 ports scanned, 982 ports were closed.
- Ports found open: 80 (http), 443 (https)

Nikto Vulnerability Scan Results:

- Number of vulnerabilities/items reported: 7
- Error limit reached: 20 errors

Nessus was found to be the most comprehensive tool, capable of identifying the highest number of vulnerabilities compared to Nmap and Nikto shown in Table 2. Nikto, although good at detecting web server vulnerabilities (over 6700 vulnerabilities), failed to detect some common vulnerabilities like SQL injection, improper error management, and denial of service shown in Table 3.

Nmap's strength lies in gathering information about ports and services running on a system, but it is not as effective as Nessus for software vulnerability scanning.

As shown in Table 3, the Nessus vulnerability analysis tool works better when compared to Nmap and Nikto. Nessus is possible to identify more vulnerability than the other tools. Even though the Nikto looks good in Table 2, it fails to detect some common vulnerabilities.

Table 2. Features of vulnerability scanners

Features	Nmap	Nessus	Nikto
Cost	It is free to download	It is not free for a long time.	It is free to use.
Usage	It can be used to access the uncontrolled part of the system	It is a security scanning tool to identify malicious files	It is useful to identify outdated web servers, software and version-specific problems.
Operating Systems	Windows and Linux	Windows, Linux, Mac, Unix	Windows, Linux
Advantage	It helps to protect the system network from intruders.	It doesn't prevent attacks; it is only used to check weaknesses.	It can detect more than 6700 vulnerabilities.
Disadvantage	It is not a better option for software	It is not a better option for networks.	It runs at the command line without any graphical user interface.

Table 3. Vulnerabilities comparison

Vulnerabilities	Nmap	Nessus	Nikto
SQL Injection	Yes	Yes	No
Improper Error Management	Yes	Yes	No
Cross-site Scripting	Yes	Yes	Yes
Denial of Service	Yes	Yes	No
Remote Code Execution	No	Yes	No

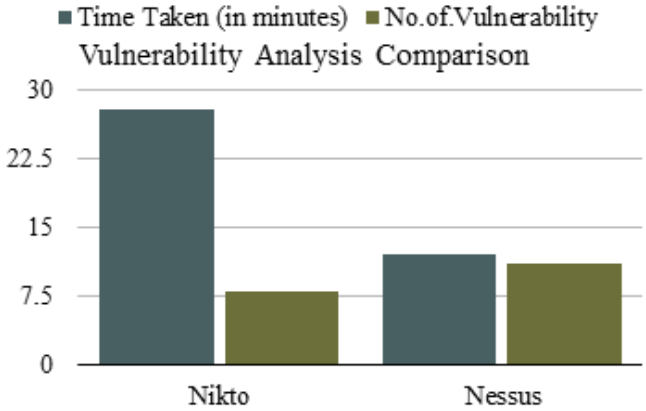

Figure 5. Scan time vs number of vulnerabilities in vulnerability analysis

Figure 5 shows each vulnerability analysis tool's time to perform a full scan on the web application. Nessus took only 18 minutes to scan and identify 37 vulnerabilities, while Nikto took 32 minutes but could only find 7 vulnerabilities. This quantitative comparison highlights Nessus as a more efficient and comprehensive vulnerability scanner. In a phishing attack, this paper concentrates on two different tools, Zphisher and Nexphisher. Both tools are open-source tools. The Nexphisher tool was developed by combining the Zphisher and HiddenEye tools. Even though Nexphisher is an advancement of Zphisher, it doesn't have some main features like custom page generation, which Zphisher works. In Zphisher, the user can create a custom webpage based on their requirement instead of using predefined templates, but in Nexphisher, that feature is not available.

Figure 6 shows that the Zphisher works better than the Nexphisher because Zphisher takes less time (7 minutes vs 12 minutes) and has more features (8 vs 5) than the Nexphisher for setting up a phishing attack. In addition to this, we also used the w3af tool, which revealed several vulnerabilities, including a wildcard DNS configuration, sensitive information disclosure, potential cross-site scripting (XSS) vulnerabilities, potential cross-site tracing (XST) vulnerability, and the presence of a deliberately vulnerable application used for testing purposes.

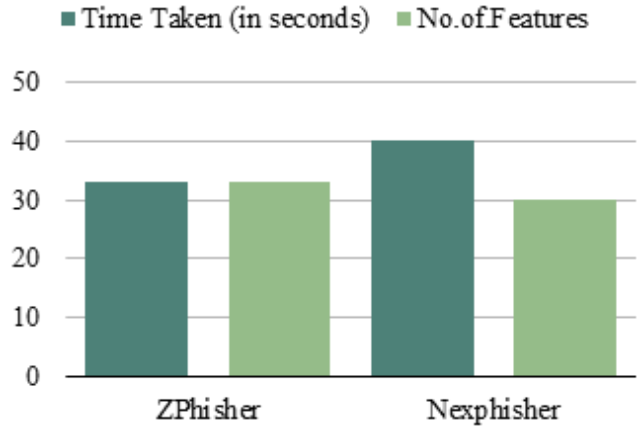

Figure 6. Scan time and number of features in phishing attack

6. CONCLUSION

The present paper delved deep into how penetration testing works and how to find those pesky vulnerabilities in web applications. It reinforced how important prevention is. Input validation, whitelisting, and all that stuff to stop attacks like XSS, SQL injection, and phishing –is super important. But it is also tricky to put into practice, especially when you're dealing with a big, complicated web application. The present paper highlighted the significance of incorporating manual testing alongside automated tools for penetration testing. Manual testing can reveal weaknesses that could be overlooked by automated scanners, particularly in intricate web applications. Looking ahead, multiple practical recommendations can be proposed to tackle the constraints and improve the efficiency of penetration testing for web applications as summarized as follows. 1) Utilizing multiple vulnerability scanning tools: 2) The importance of continuous research and development in penetration testing methodologies and techniques is emphasized 3) The importance of collaboration between development and security teams throughout the software development process is suggested, and 4) Advise on conducting regular security audits and assessments, which should include penetration testing, to consistently assess the web application's security state and detect any new vulnerabilities that may have occurred because of updates or upgrades. By acknowledging these constraints and putting into practice the proposed suggestions, companies can improve their capacity to pre-emptively detect and address weaknesses in web applications, thoroughly safeguarding their important resources and guaranteeing the security, reliability, and accessibility of their systems and data.

REFERENCES

[1] Goutam, A., Tiwari, V. (2019). Vulnerability assessment and penetration testing to enhance the security of web application. In 2019 4th International Conference on Information Systems and Computer Networks, Mathura, India, pp. 601-605. https://doi.org/10.1109/ISCON47742.2019.9036175

[2] Nagpure, S., Kurkure, S. (2017). Vulnerability assessment and penetration testing of web application. In 2017 International Conference on Computing, Communication, Control and Automation, Pune, India, pp. 1-6. https://doi.org/10.1109/ICCUBEA.2017.8463920

[3] Kothamasu, G.A., Venkata, S.K.A., Pemmasani, Y., Mathi, S. (2023). An Investigation on Vulnerability Analysis of Phishing Attacks and Countermeasures. International Information and Engineering Technology Association, 13(2): 333-340. https://doi.org/10.18280/ijsse.130215

[4] Nakkeeran, M., Mathi, S. (2021). A generalized comprehensive security architecture framework for IoT applications against cyber-attacks. In Artificial Intelligence and Technologies, vol 806. Springer, Singapore, pp. 455-471. https://doi.org/10.1007/978-981-16-6448-9_46

[5] Shebli, H.M.Z.A., Beheshti, B.D. (2018). A study on penetration testing process and tools. In 2018 IEEE Long Island Systems, Applications and Technology Conference, Farmingdale, NY, USA, pp. 1-7. https://doi.org/10.1109/LISAT.2018.8378035

[6] Pandey, R., Jyothindar, V., Chopra, U.K. (2020). Vulnerability assessment and penetration testing: a portable solution Implementation. In 2020 12th International Conference on Computational Intelligence and Communication Networks, Bhimtal, India, pp. 398-402. https://doi.org/10.1109/CICN49253.2020.9242640

[7] Ojagbule, O, Wimmer, H, Haddad, R.J. (2018). Vulnerability analysis of content management systems to SQL injection using SQLMAP. In SoutheastCon2018, St. Petersburg, FL, USA, pp. 1-7. https://doi.org/10.1109/SECON.2018.8479130

[8] Fadlalla, F.F., Elshoush, H.T. (2023). Input validation vulnerabilities in web applications: Systematic review, classification, and analysis of the current state-of-the-art. IEEE Access, 11: 40128-40161. https://doi.org/10.1109/ACCESS.2023.3266385

[9] Aibekova, A., Selvarajah, V. (2022). Offensive security: Study on penetration testing attacks, methods, and their types. In 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, Ballari, India, pp. 1-9. https://doi.org/10.1109 ICDCECE53908.2022.9792772

[10] Arnaldy, D., Perdana, A.R. (2019). Implementation and analysis of penetration techniques using the man-in-the-middle attack. In 2019 2nd International Conference of Computer and Informatics Engineering, Banyuwangi, Indonesia, pp. 188-192. https://doi.org/10.1109/IC2IE47452.2019.8940872

[11] Cisar, P., Pinter, R. (2019). Some ethical hacking possibilities in Kali Linux environment. Journal of Applied Technical and Educational Sciences, 9(4): 129-149. https://doi.org/10.24368/jates.v9i4.139

[12] Devi, R.S., Kumar, M.M. (2020). Testing for security weakness of web applications using ethical hacking. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI) (48184), Tirunelveli, India, pp. 354-361. https://doi.org/10.1109/ICOEI48184.2020.9143018

[13] Anand, P., Singh, A.S. (2021). Penetration testing security tools: A comparison. In 2021 10th International Conference on System Modeling & Advancement in Research Trends, MORADABAD, India, pp. 182-184. https://doi.org/10.1109/SMART52563.2021.9676283

[14] Karayat, R., Jadhav, M., Kondaka, L.S., Nambiar, A. (2022). Web application penetration testing & patch development using Kali Linux. In 2022 8th International Conference on Advanced Computing and Communication Systems, Coimbatore, India, pp. 1392-1397. https://doi.org/10.1109/ICACCS54159.2022.9785232

[15] Kandasamy, K., Srinivas, S., Achuthan, K., Rangan, V. P. (2022). Digital healthcare-cyberattacks in asian organizations: an analysis of vulnerabilities, risks, nist perspectives, and recommendations. IEEE Access, 10: 12345-12364. https://doi.org/10.1109/ACCESS.2022.3145372

[16] Gunawan, T.S., Lim, M.K., Kartiwi, M., Malik, N.A., Ismail, N. (2018). Penetration testing using Kali linux: SQL injection, XSS, wordpres, and WPA2 attacks. Indonesian Journal of Electrical Engineering and Computer Science, 12(2): 729-737. https://doi.org/10.11591/ijeecs.v12.i2.pp729-737

[17] Kumar, R., Tlhagadikgora, K. (2018). Internal network penetration testing using free/open source tools: Network and system administration approach. Advanced Informatics for Computing Research, 257-269. https://doi.org/10.1007/978-981-13-3143-5_22

[18] Karangle, N., Mishra, A.K., Khan, D.A. (2019). Comparison of Nikto and Uniscan for measuring URL vulnerability. In 2019 10th International Conference on Computing, Communication and Networking Technologies, Kanpur, India, pp. 1-6. https://doi.org/10.1109/ICCCNT45670.2019.8944463

[19] Visoottiviseth, V., Kotarasu, C., Cheunprapanusorn, N., Chamornmarn, T. (2019). A mobile application for security assessment towards the internet of thing devices. In 2019 IEEE 6th Asian Conference on Defence Technology, Bali, Indonesia, pp. 1-7. https://doi.org/10.1109/ACDT47198.2019.9072921

[20] Shah, M., Ahmed, S., Saeed, K., Junaid, M., Khan, H. (2019). Penetration testing active reconnaissance phase–optimized port scanning with Nmap tool. In 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies, Sukkur, Pakistan, pp. 1-6. https://doi.org/10.1109/ICOMET.2019.8673520

[21] Nathezhtha, T., Sangeetha, D., Vaidehi, V. (2019). WC-PAD: web crawling based phishing attack detection. In 2019 International Carnahan Conference on Security Technology, Chennai, India, pp. 1-6. https://doi.org/10.1109/CCST.2019.8888416

[22] Junmei, W., Chengkang, Y. (2021). Automation testing of software security based on burpsuite. In 2021 International Conference of Social Computing and Digital Economy, Chongqing, China, pp. 71-74. https://doi.org/10.1109/ICSCDE54196.2021.00025

[23] Bherde, G.P., Pund, M.A. (2016). Recent attack prevention techniques in web service applications. In

2016 International Conference on Automatic Control and Dynamic Optimization Techniques, Pune, India, pp. 1174-1180. https://doi.org/10.1109/ICACDOT.2016.7877771

[24] Nischitha, G.K., Sahana, S., Santhosh Kumar, B.J. (2019). Detection and avoidance of web vulnerability using XSS. International Journal of Recent Technology and Engineering, 8(2): 1737-1740. https://doi.org/10.35940/ijrte.B1039.078219

[25] Abed, T.M., Abdul-Wahab, H.B. (2019). Anti-phishing system using intelligent techniques. In 2019 2nd Scientific Conference of Computer Sciences, Baghdad, Iraq, pp. 44-50. https://doi.org/10.1109/SCCS.2019.8852601

[26] Swarup, Y., Kumar, A., Tyagi, A., Kumar, V. (2021). Prevention of SQL injection attacks using query hashing technique. In 2021 2nd International Conference on Range Technology, Chandipur, Balasore, India, pp. 1-5. https://doi.org/10.1109/ICORT52730.2021.9581804

[27] Adil, M., Khan, R., Ghani, M.A.N.U. (2020). Preventive techniques of phishing attacks in networks. In 2020 3rd International Conference on Advancements in Computational Sciences, Lahore, Pakistan, pp. 1-8. https://doi.org/10.1109/ICACS47775.2020.9055943