Suceava CTF

Author: MTA 2

Bobeanu Catalina/Brinzea Andrei/Cujba Mihai/Lungu Andrei

/* la final sunt puse utilitarele de enumerare+modul in care le-am folosit */

Brazil:

Summary

< Brazil >
Proof of Flag
Summary
Proof of Solving

Proof of Flag

Brazil: 0e954f564b7aef949c65b8dc6cac8208

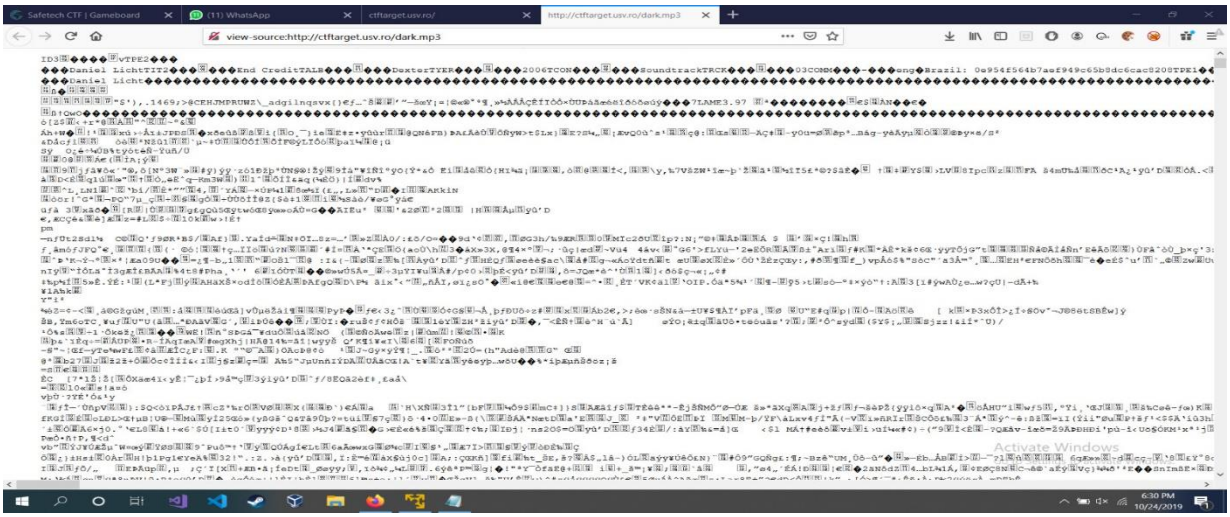
Summary

Am primit ca indiciu de pe prima pagina a site-ului sa fim atenti la sunet. Am cautat sa vad ce mai exact este acel sunet, iar cand am intrat int stringuri am gasit flag-ul.

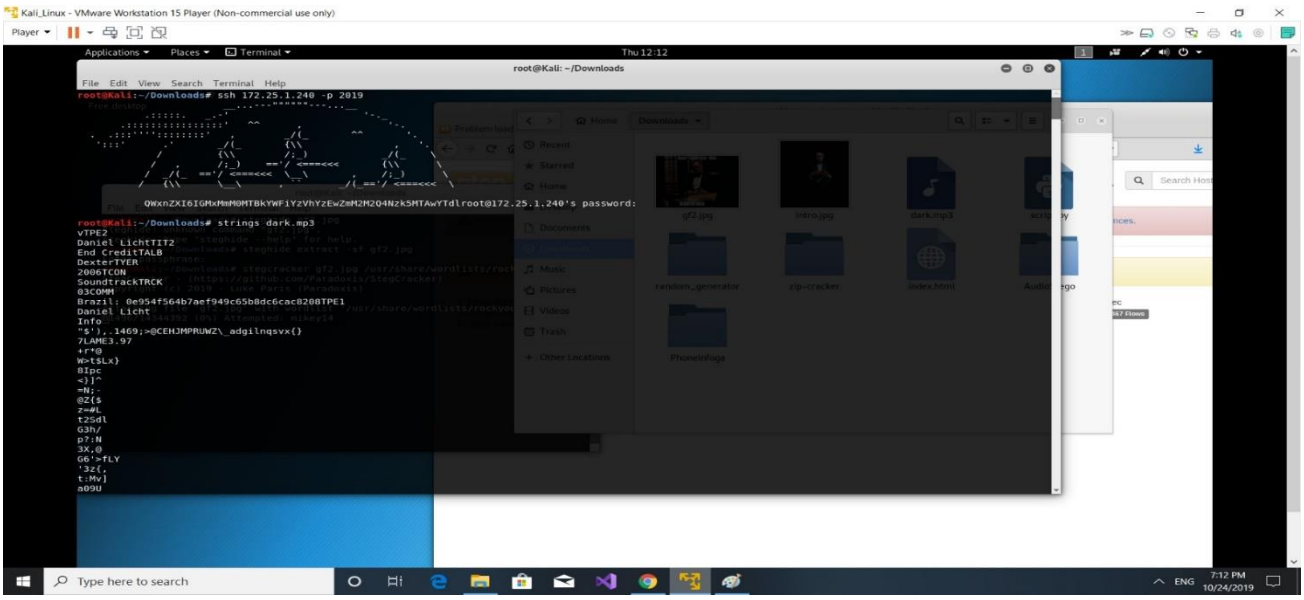
Proof of Solving

Am dat view-source pe prima pagina, apoi am vazut ca acolo se afla un fisier mp3 dark.mp3, pe care l-am inspectat la randul lui si am scos flag-ul. Un coleg de echipa a downloadat mp3-ul si a folosit comanda strings pe el pentru a scoate hash-ul.

*Varianta cu view-source



*Varianta cu strings (include si o parte din challenge-ul de la Algeria)



Alger:

Summary

< Alger >

Proof of Flag

Summary

Proof of Solving

Proof of Flag

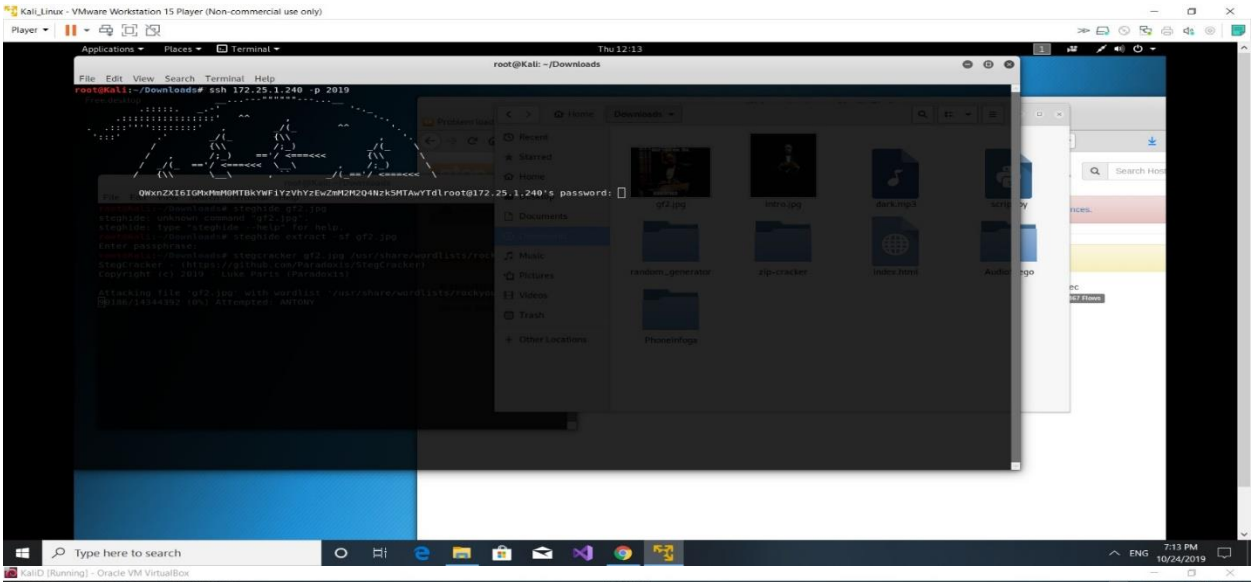
Alger: c12c410daabc5ac10fc63d8799100a7e

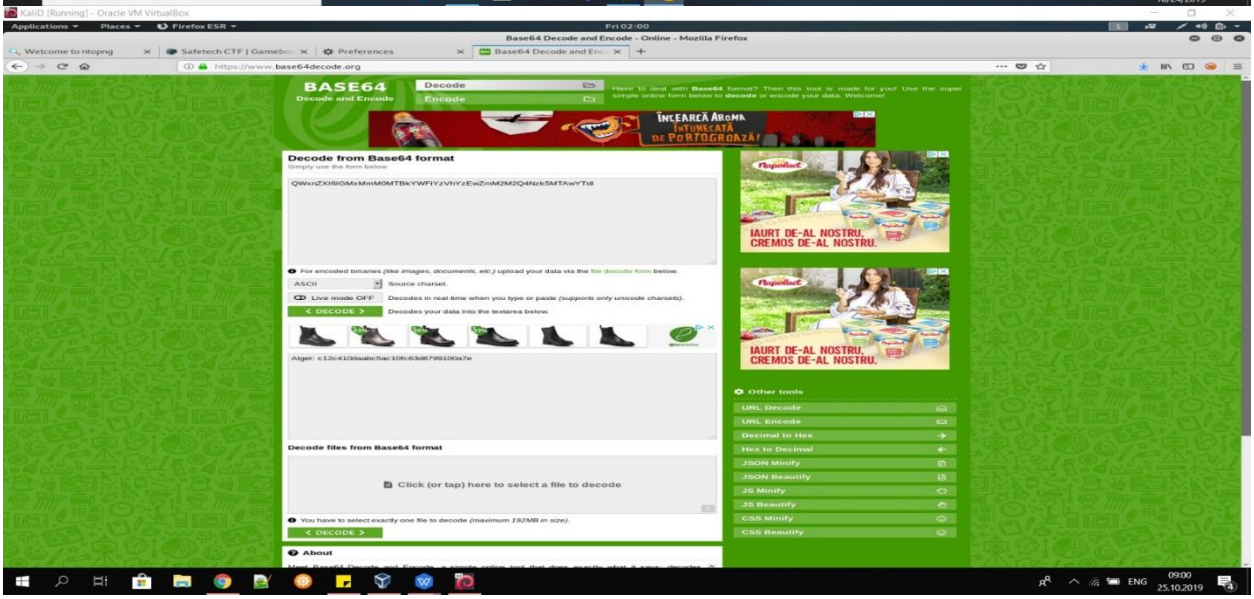
Summary

Am gasit un port de ssh deschis, unde se afla un base64 din care se scotea flag-ul.

Proof of Solving

Am enumerate toate porturile de pe masina, iar unul din cele care au iesit in evidenta a fost un ssh deschis, pe portul 2019. Ne-am conectat prin ssh la ip si port, iar inainte sa ne ceara credentialele de logare a fost printata o poza cu niste vrajitoare, iar sub un base64. L-am luat, l-am decodificat si am scos flag-ul.





Portugal:

Summary

< Portugal >

Proof of Flag

Summary

Proof of Solving

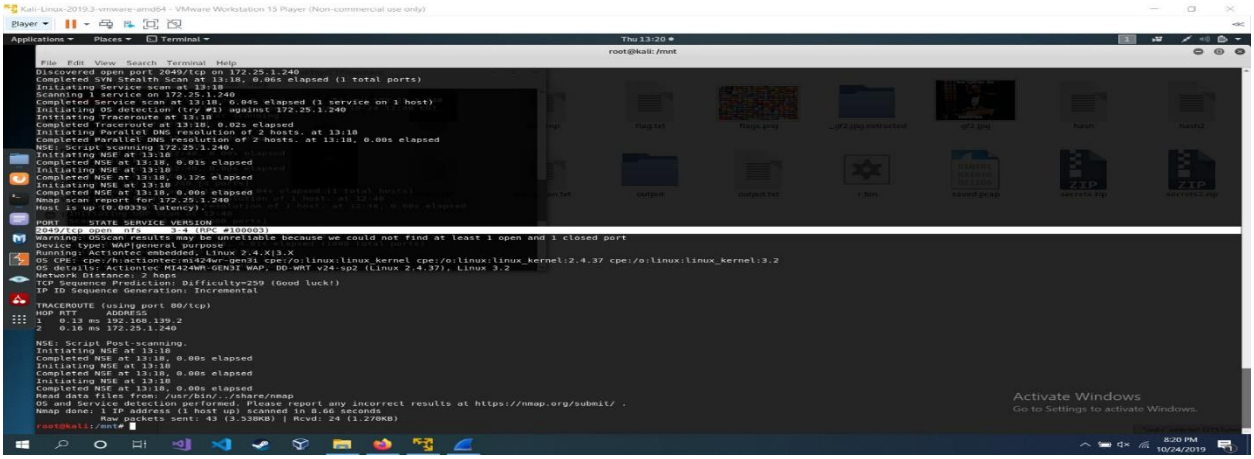
Proof of Flag

Portugal: f0b501295389923de36e398b93d2832f

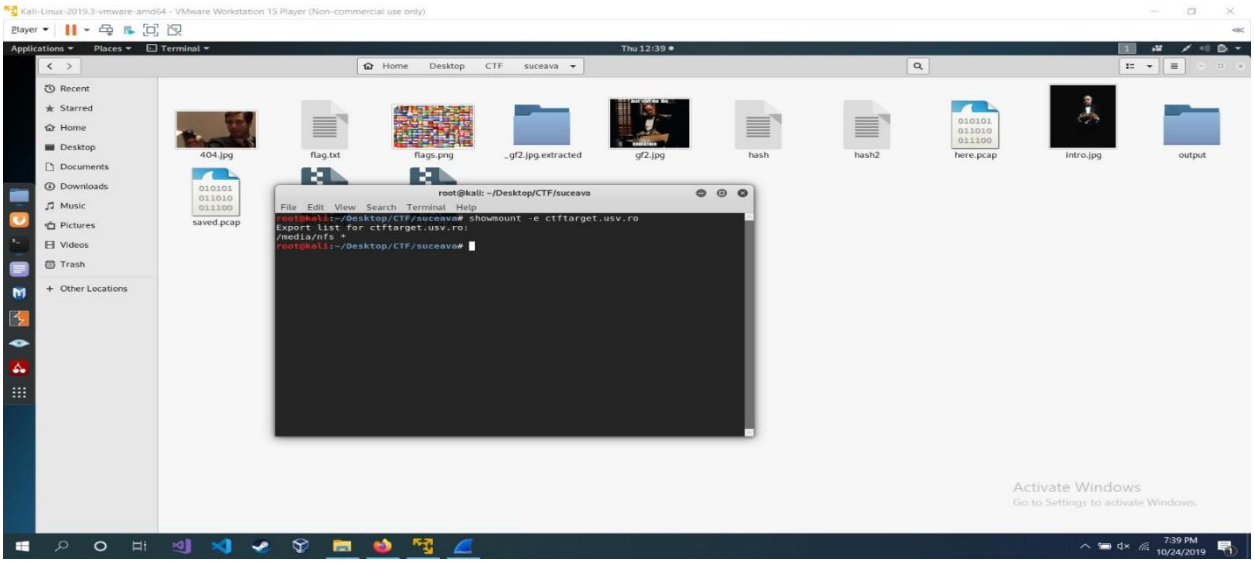
Summary

Am gasit un port deschis pe serviciul nfs. Impreuna cu acesta erau mai multe porturi deschise astfel incat am putut sa montez un sistem de fisiere de pe target pe masina locala.

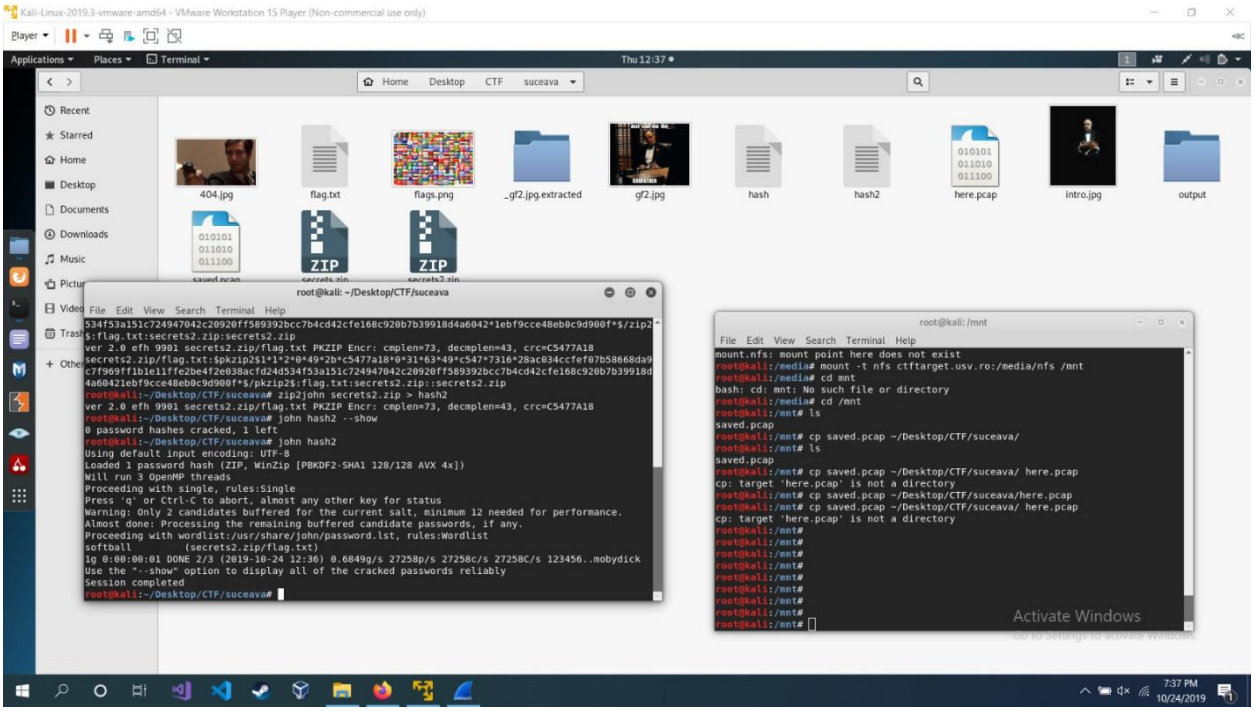
Proof of Solving



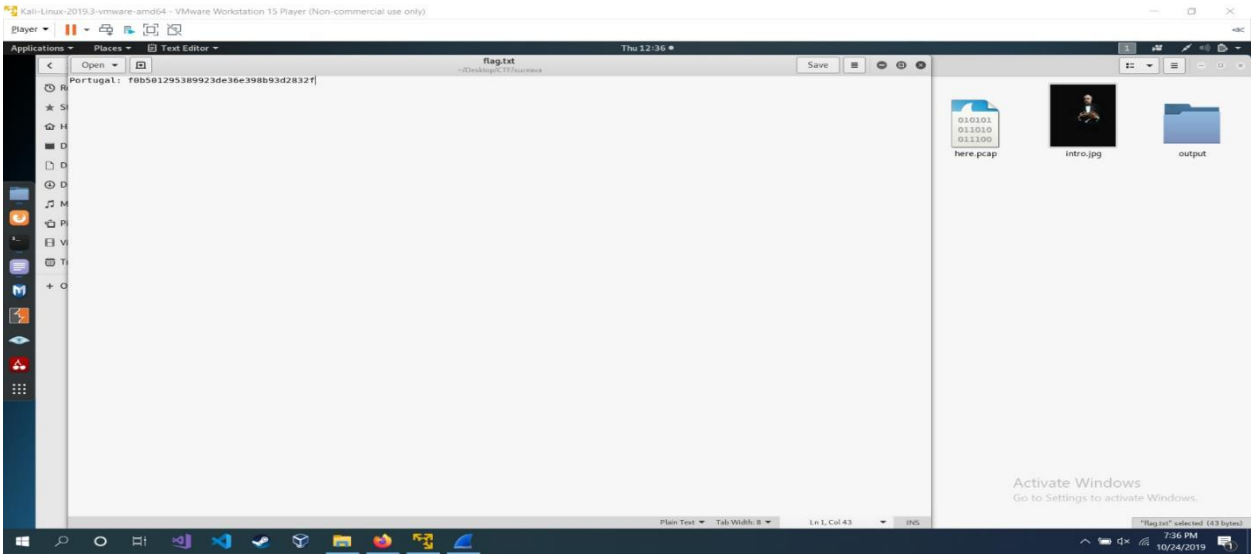
Am verificat prin comanda showmount -e ctftarget.usv.ro ce system de fisiere poate fi exportat de pe masina target si montat pe masina mea locala.



Odata gasit sistemul de fisiere m-am folosit de utilitarul mount pentru a monta sistemul pe masina locala.



Dupa ce am montat sistemul si am vazut ca in el se afla o captura de wireshark am cautat in ea si am vazut niste traffic pe http. Am descarcat de acolo 2 poze care de fapt erau 2 zip-uri, iar in ele se afla flag-ul. Am folosit utilitarul JohnTheRipper pentru a sparge parola de la zip (softball) si de aici am scos flag-ul.



Columbia:

Summary

< Columbia >

Proof of Flag

Summary

Proof of Solving

Proof of Flag

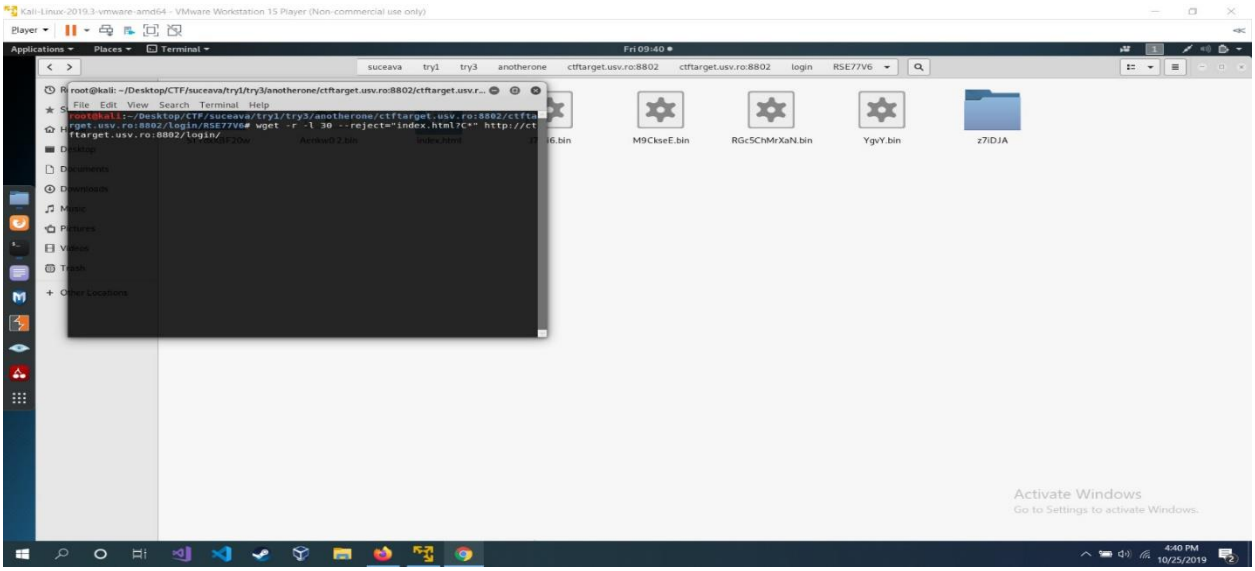
Columbia: c988e5e5b6970c52854826affd29679a

Summary

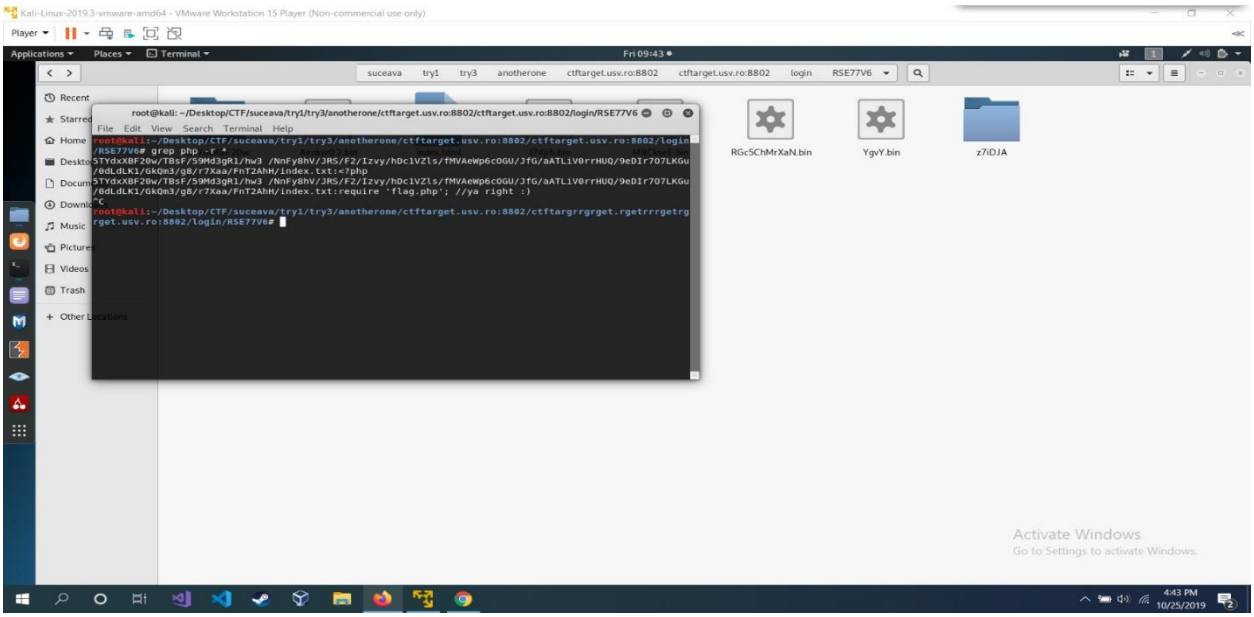
Am gasit un port deschis, respective portul 8802. Nu spunea prea multe, asa ca am intrat pur si simplu pe browser pe el. Acolo am gasit foarte multe directoare cu foarte multe subdirectoare facute recursive.

Proof of Solving

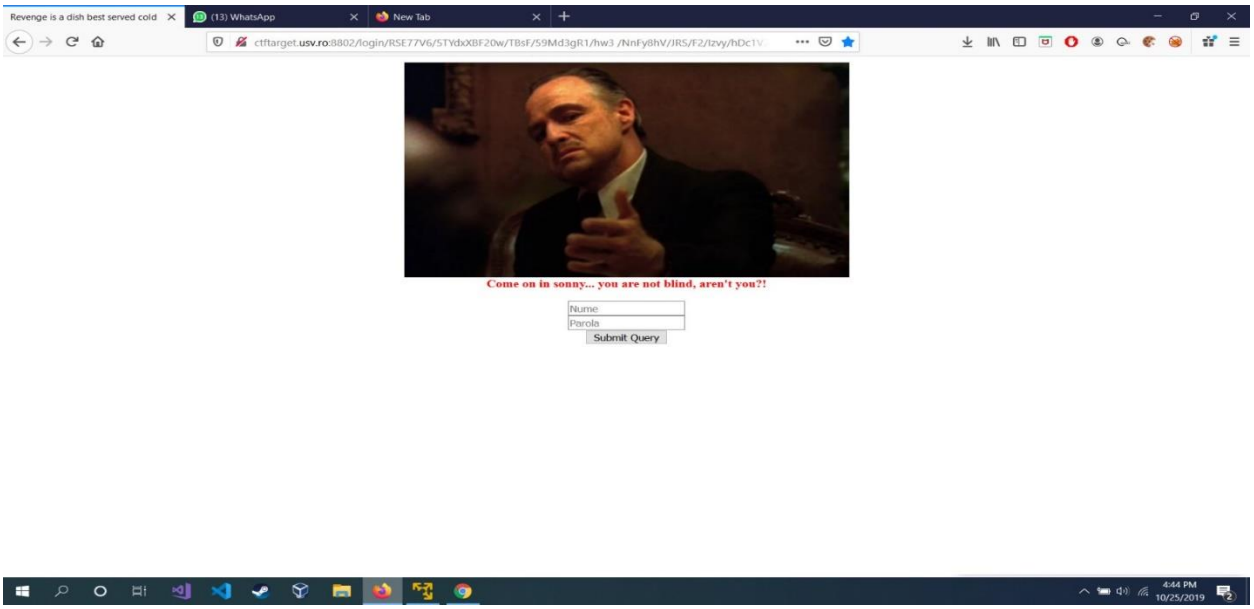
Primul lucru pe care l-am facut cand am vazut atatea directoare a fost sa intru prin ele, in idea ca voi gasi ceva. In unele dintre ele se gaseau fisiere binare fara niciun sens. Atunci m-am gandit ca ar fi mai bine sa caut un pattern. M-am gandit ca intr-unul din binare se gaseste de fapt un ELF si de fapt e un challenge de Pwn. Asa ca am descarcat o foarte mare bucata din fisierele (~117.000) de pe portul respectiv, in speranta cavoi gasi ceva.



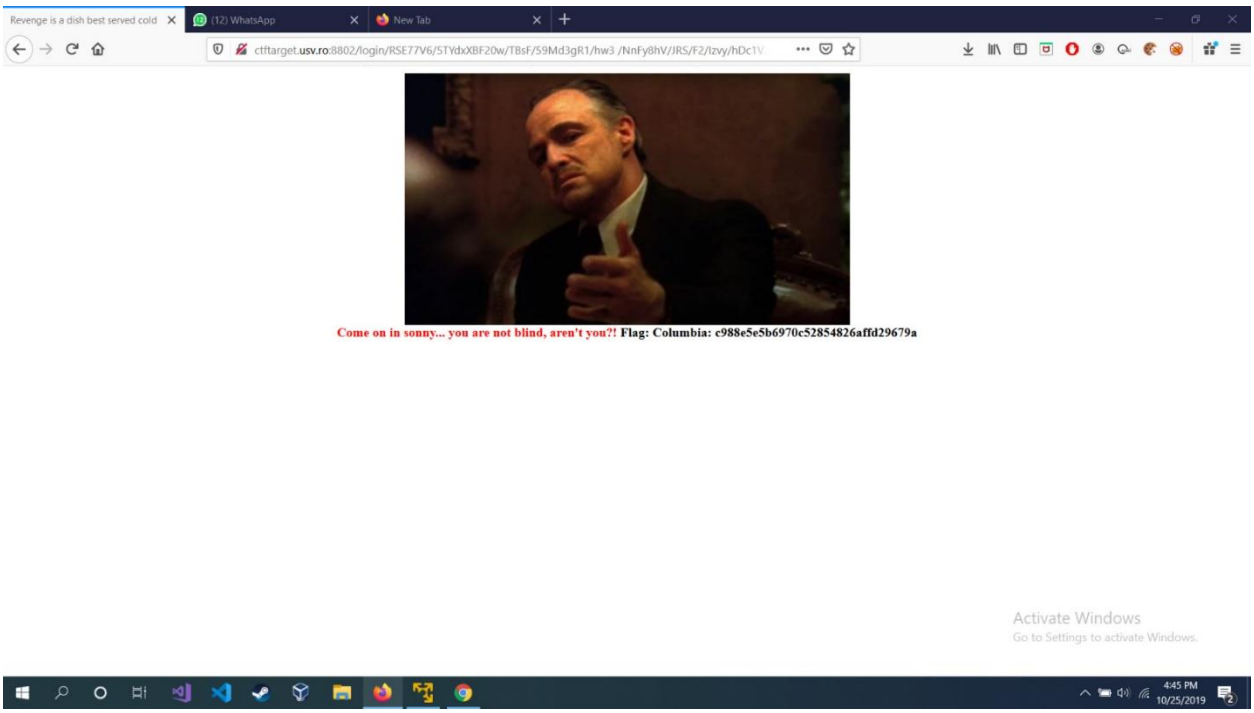
Mai departe am cautat cu grep ELF -r *, dar faraefect. Apoi am vazut ca odata cu binarele acelea am si paginile index.html de la fiecare subdirector in parte. Prima data am cautat dupa vre-un cod javascript ascuns, dar nu am gasit nimic interesant si m-am gandit sa caut dupa vre-un script php. Aici am gasit ceva interesant.



Mi-a sarit in ochi “flag.php”. Am intratm pe pagina inapoi, m-am dus pana la subdirectorul respective si am gasit asta:



O pagina de log-in in care user-ul si parola hash-uite in md5 sa aiba aceeasi hash. Aveam de ales intre a cauta 2 stringuri care sa faca o coliziunde de md5, slabe sanse, sau sa ma folosesc de faptul ca era vorba totusi de php. M-am folosit de vulnerabilitatea php-ului, respectiv juggling type, punand numele si parola ca vectori, nu ca stringuri, dar diferite, pentru a respecta toate conditiile. Iar asa am scos flag-ul.



Aparent nu se vede payload-ul de juggling type, iar acum nu mai am acces la platforma, dar il voi pune aici: url/?nume[]=&parola[]=1.

Morrocco:

Summary

< Morrocco >
Proof of Flag
Summary
Proof of Solving

Proof of Flag

Morrocco: 2eef81d7c3ae900e09b18f266a50db70

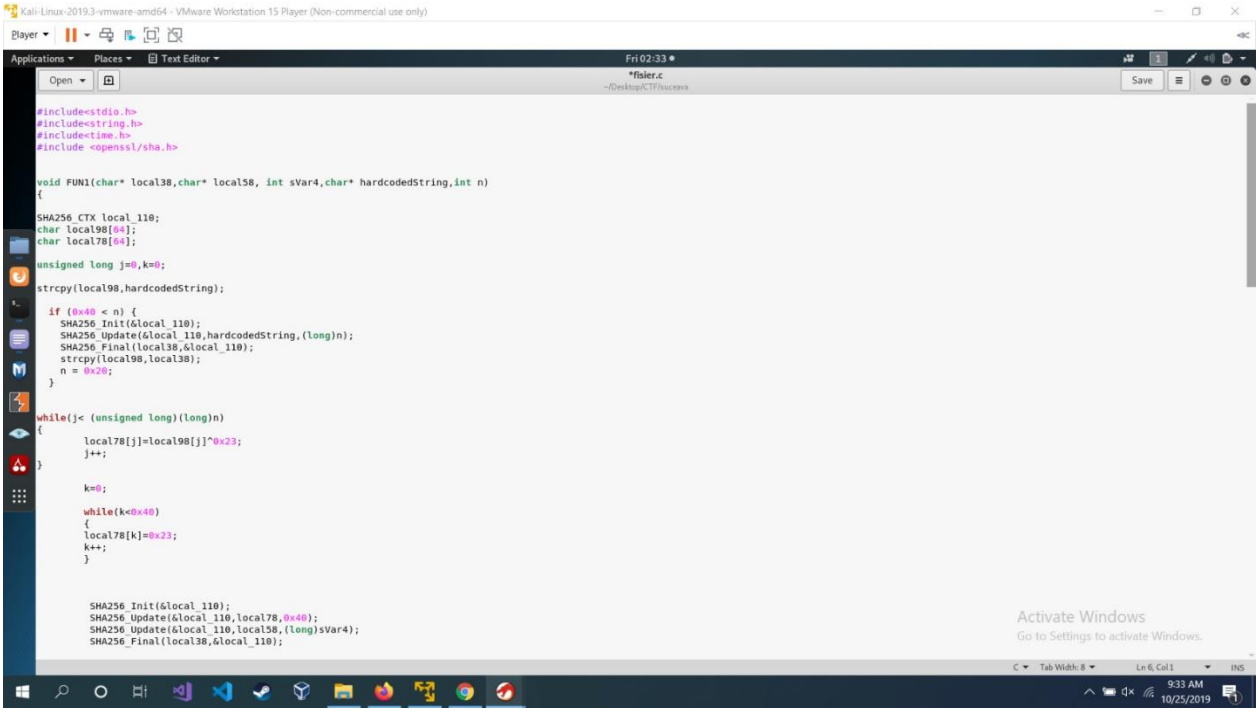
Summary

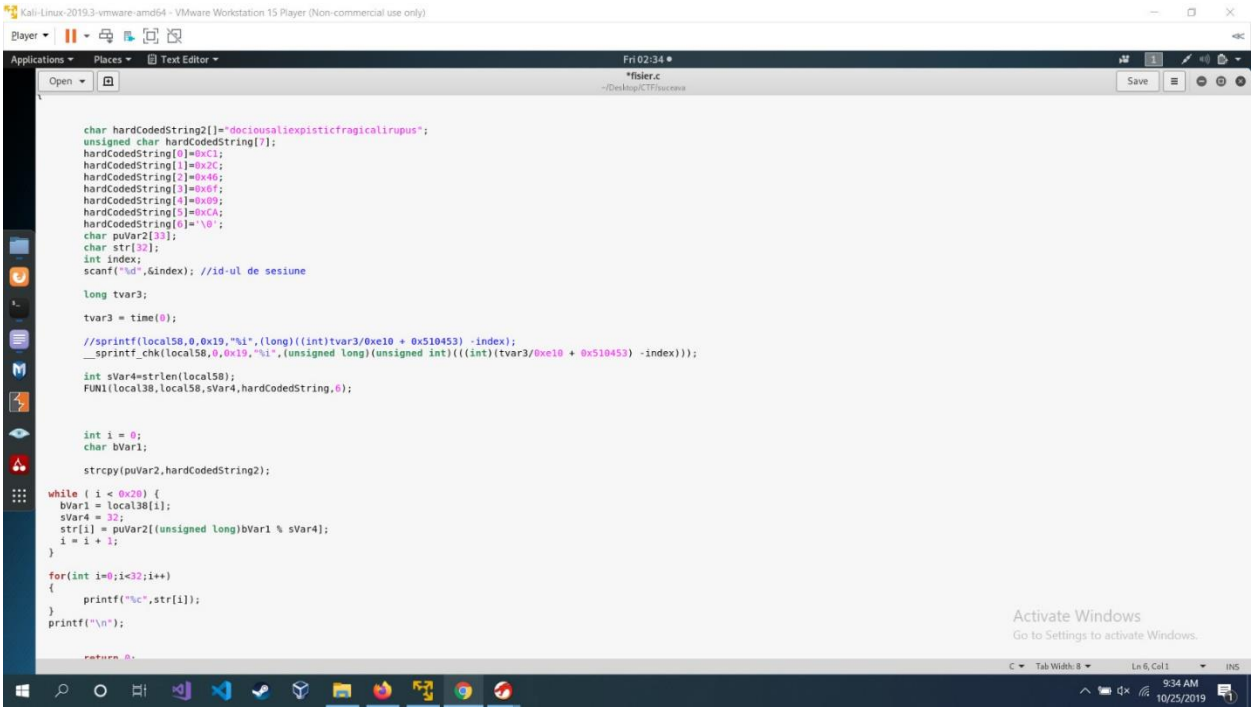
Aici am gasit in prima faza un port deschis 8888. Am incercat prima data sa ma conecteze pe browser si nu se prea intampla mare lucru, mi se spunea doar ca m-am conectat cu un ID si dupa "Access Denied". Dupa asta m-am gandit sa incerc cu netcat, iar asa am vazut ca de fapt in spate rula un program ce astepta un input de la tastatura, iar in functie de acesta luam sau nu flag-ul.

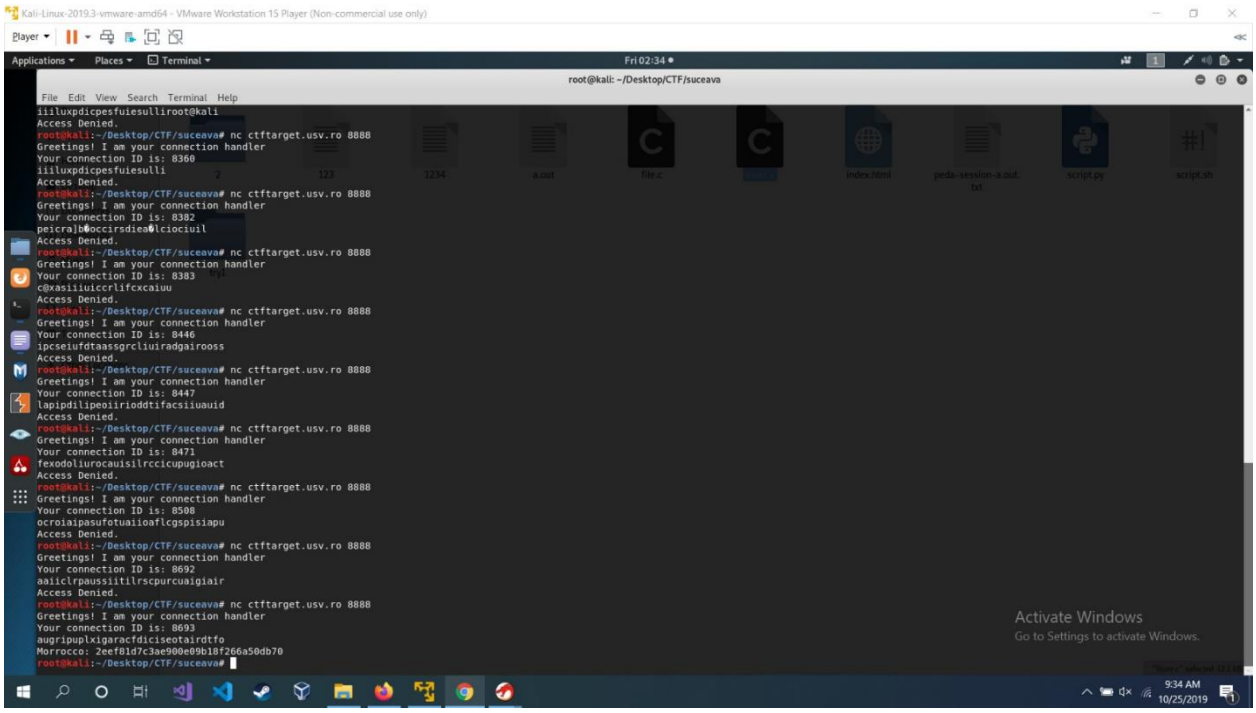
Proof of Solving

Odata cu scanarea tuturor porturilor am observant ca este un server de ftp deschis pe portul 2, cu login anonymous. Asta inseamna ca ma puteam loga cu userul anonymous si fara parola. Am intrat, am gasit acolo un binary numit server, pe care l-am descarcat pe masina locala si am inceput sa inteleg ce se petrece. In spatele portului 8888 era de fapt acel binary de MacOS. L-am luat si l-am analizat cu Ghidra. De acolo se vedea ca genereaza un ID, dupa care creea un thread prin care genera de fapt stringul pe care il astepta ca input. Avea niste functii de sha256, deci era cam greu s ail pacalesti, avand in vedere ca era vorba de un hash aici.

Am deschis un editor de text si m-am apucat sa refac codul in C pentru binar, in idea ca voi rula si eu la randul meu programul, dar cu inputul potrivit, respectiv ID-ul pe care il voi primi de la server si voi scoate flag-ul. L-am refacut, dar Ghidra mai foloseste si functii custom precum SEXT48, iar aici m-am blocat putin. Am intrat in IDA sa vad ce se intampla mai exact acolo, dar singurul lucru era faptul ca lua ca punct de inceput pentru un for un anume RBP+n, ceea ce insemna ca, strict in programul nostru, se continua creearea unui string dupa ce primii n octeti au fost modificati(mai exact printr-un xor).


Kali-Linux-2019.3-vmware-amd64 - VMware Workstation 15 Player (Non-commercial use only)
Player
Applications
Places
Text Editor
Fri 02:33
Open
*fisier.c
~/Desktop/CTF/succava
Save
#include<stdio.h>
#include<string.h>
#include<time.h>
#include <openssl/sha.h>
void FUN1(char* local38,char* local58, int sVar4,char* hardcodedString,int n)
{
SHA256_CTX local_110;
char local98[64];
char local78[64];
unsigned long j=0,k=0;
strcpy(local98,hardcodedString);
if (0x40 < n) {
SHA256_Init(&local_110);
SHA256_Update(&local_110,hardcodedString,(long)n);
SHA256_Final(local38,&local_110);
strcpy(local98,local38);
n = 0x20;
}
while(j< (unsigned long)(long)n)
{
local78[j]=local98[j]^0x23;
j++;
}
k=0;
while(k<0x40)
{
local78[k]=0x23;
k++;
}
SHA256_Init(&local_110);
SHA256_Update(&local_110,local78,0x40);
SHA256_Update(&local_110,local58,(long)sVar4);
SHA256_Final(local38,&local_110);
Activate Windows
Go to Settings to activate Windows.
C
Tab Width: 8
Ln 6, Col 1
INS
9:33 AM
10/25/2019


Kali-Linux-2019.3-vmware-amd64 - VMware Workstation 15 Player (Non-commercial use only)
Player
Applications
Places
Text Editor
Fri 02:34
Open
*fisier.c
~/Desktop/CTF/succava
Save
char hardCodedString2[]="dociousaliexpisticfragicalirupus";
unsigned char hardCodedString[7];
hardCodedString[0]=0xC1;
hardCodedString[1]=0x2C;
hardCodedString[2]=0x46;
hardCodedString[3]=0x6f;
hardCodedString[4]=0x09;
hardCodedString[5]=0xCA;
hardCodedString[6]='\0';
char puVar2[33];
char str[32];
int index;
scanf("%d",&index); //id-ul de sesiune
long tvar3;
tvar3 = time(0);
//sprintf(local58,0,0x19,"%i",(long)((int)tvar3/0xe10 + 0x510453) -index);
__sprintf_chk(local58,0,0x19,"%i",(unsigned long)(unsigned int)(((int)(tvar3/0xe10 + 0x510453) -index)));
int sVar4=strlen(local58);
FUN1(local38,local58,sVar4,hardCodedString,6);
int i = 0;
char bVar1;
strcpy(puVar2,hardCodedString2);
while (i < 0x20) {
bVar1 = local38[i];
sVar4 = 32;
str[i] = puVar2[(unsigned long)bVar1 % sVar4];
i = i + 1;
}
for(int i=0;i<32;i++)
{
printf("%c",str[i]);
}
printf("\n");
Activate Windows
Go to Settings to activate Windows.
C
Tab Width: 8
Ln 6, Col 1
INS
9:34 AM
10/25/2019



Atasez aici codul in C:

```
 #include<stdio.h>

#include<string.h>

#include<time.h>

#include <openssl/sha.h>


void FUN_100001990(unsigned char *puParm1,void *pvParm2,int iParm3,unsigned char *puParm4,int iParm5)


{

 unsigned long local130;

 unsigned long l;

 unsigned long k;

 unsigned long j;

 SHA256_CTX local110;

 int n;

 unsigned char *local98;

 char local78 [64];

 unsigned char local38 [40];

 long local10;

 n = iParm5;
```

```
local98 = puParm4;
if (0x40 < iParm5) {
  n = iParm5;
  SHA256_Init(&local110);
  SHA256_Update(&local110,puParm4,(long)n);
  SHA256_Final(local38,&local110);
  local98 = local38;
  n = 0x20;
}
j = 0;
while (j < (unsigned long)(long)n) {
  local78[j] = local98[j] ^ 0x23;
  j = j + 1;
}
k = n;
while (k < 0x40) {
  local78[k] = 0x23;
  k = k + 1;
}
SHA256_Init(&local110);
SHA256_Update(&local110,local78,0x40);
SHA256_Update(&local110,pvParm2,(long)iParm3);
SHA256_Final(puParm1,&local110);
l = 0;
while (l < (unsigned long)(long)n) {
  local78[l] = local98[l] ^ 0x3c;
  l = l + 1;
}
local130 = n;
while (local130 < 0x40) {
  local78[local130] = 0x3c;
  local130 = local130 + 1;
}
SHA256_Init(&local110);
SHA256_Update(&local110,local78,0x40);
```

```
 SHA256_Update(&local110,puParm1,0x20);
 SHA256_Final(puParm1,&local110);
}



int main()
{
 char hardCodedString2[]="dociousaliexpisticfragicalirupus";
                unsigned char hardCodedString[7];
                hardCodedString[0]=0xC1;
                hardCodedString[1]=0x2C;
                hardCodedString[2]=0x46;
                hardCodedString[3]=0x6f;
                hardCodedString[4]=0x09;
                hardCodedString[5]=0xCA;
                hardCodedString[6]='\0';
 char aaa[32];
 int x=1;
 scanf("%d",&x);
 char bVar1;
 char *puVar2;
 long long tVar3;
 size_t sVar4;
 int i;
 char local_58 [32];
 char local_38 [40];
 long local_10;
 tVar3 = time(0);
 __sprintf_chk(local_58,0,0x19,"%i",(unsigned long)(unsigned int)(((int)(tVar3 / 0xe10) + 0x510453) - x));
 sVar4 = strlen(local_58);
 FUN_100001990(local_38,local_58,sVar4,hardCodedString,6);
 i = 0;
 while (i < 0x20) {
   bVar1 = local_38[(long)i];
```

```
  sVar4 = strlen(hardCodedString2);
  aaa[(long)i] = hardCodedString2[(unsigned long)bVar1 % sVar4];
  i = i + 1;
 }
 aaa[32]='\0';
 printf("%s\n",aaa);
}
```

UTILITARE FOLOSITE PENTRU ENUMERARE:

nmap:

- Pentru enumerarea tuturor porturilor am folosit comanda nmap -v -A -T4 -p- 172.25.1.240
- Pentru enumerarea unui anume port am folosit nmap -v -A -T4 -pport 172.25.1.240
- Pentru enumerarea porturilor UDP am folosit nmap -v -A -T4 -sU 172.25.1.240

Dirb/gobuster

- Le-am folosit pe ambele, dirb pentru enumerarea directoarelor, iar gobuster pentru a cauta fisiere cu extensii sau pentru a ma folosi de facilitatea de thread-uri
- Dirb: dirb http://ctftarget.usv.ro:8888
- Gobuster: gobuster dir -u http://ctftarget.usv.ro:8802 -w /usr/share/wordlists/dirb/small.txt -t 20 -x .php,.txt,.html