

# RST CTF #1

Write-Up(Tiz)

# Biletul criptat

În acest challenge am avut un mesaj criptat prin mai multe metode.

Prima etapa a fost sa decodez din "brainfuck language"-am folosit

<https://www.dcode.fr/brainfuck-language>

The screenshot shows the dCode website's interface for the Brainfuck language. At the top left, there's a search bar with the text "Search for a tool" and a search button. Below it, a search bar contains the text "e.g. type 'caesar'". To the right of the search bar, there's a button that says "BROWSE THE FULL DCODE TOOLS LIST".

In the center, there's a section titled "Brainfuck" with a sub-header "Informatics > Programming Language > Brainfuck". Below this, there's a sponsored ad for "Bitdefender EDR" with the text "Is EDR right for you? Take a short assessment.".

Below the ad, there's a section titled "BRAINFUCK INTERPRETER". It contains a text input field with the code "BJCUMR33G33XC0LXMZXGM4B0GJ4GM6L0MZYHKH3Y0FQW42DUG04XGNDPN8YHC6DXIV60U---PRET". Below the input field, there's a button labeled "EXECUTE".

On the right side, there's a "Summary" section with a list of links: "Brainfuck Interpreter", "Brainfuck Encoder", "How to encrypt using Brainfuck code?", "How to encrypt using Brainfuck Shortcut code?", "How to decrypt Brainfuck code?", "How to decrypt Brainfuck Shortcut code?", "How to recognize Brainfuck coded text?", "What are the variants of the Brainfuck code?", and "When Brainfuck have been invented?".

At the bottom left, there's a banner for "REDMI 9C" with the text "PROFITĂ DE REDUCERILE MI".

# Biletul criptat

A doua etapă a constat în a decode mesajul din Base32.

[https://gchq.github.io/CyberChef/#recipe=From\\_Base32\('A-Z2-7%3D',false\)&input=QkpDVU1SMzNHSjNYQ09MWE1aWEdNNEJRR0o0R002TE9NWIIIS00zWU9GUVc0MkRVR1E0WEdORFBOQIIIQzZEWE5WNIFVPT09](https://gchq.github.io/CyberChef/#recipe=From_Base32('A-Z2-7%3D',false)&input=QkpDVU1SMzNHSjNYQ09MWE1aWEdNNEJRR0o0R002TE9NWIIIS00zWU9GUVc0MkRVR1E0WEdORFBOQIIIQzZEWE5WNIFVPT09)

În ultima etapă am folosit ROT13 pentru a ajunge la forma finală

[https://gchq.github.io/CyberChef/#recipe=ROT13\(true,true,false,13\)&input=RUZH ezJ3cTI3Zm5mcDAyeGZ5bmZwdTN4cWFuaHQ0OXM0b2hwcXh3bX0](https://gchq.github.io/CyberChef/#recipe=ROT13(true,true,false,13)&input=RUZH ezJ3cTI3Zm5mcDAyeGZ5bmZwdTN4cWFuaHQ0OXM0b2hwcXh3bX0)

Flag : **RST{2jd9jsasc02kslasch3kdnaug49f4bucdkjz}**

# Strabunicul lui Cezar

Analizând metoda prin care a fost criptat mesajul putem concluziona că s-a folosit “Atbash Cipher”

<https://www.dcode.fr/cipher-identifier>

Aplicând cifrul aflat, putem extrage mesajul în plain text

Felicitari, flag-ul este "atbash".

dar criptat. Vezi instructiunile din

Stegano II pentru mai multe detalii.

Flag:

RST{241f143d128e232349e2bf212ec0b123faff7b85944bbe1e55722b35c9207c13}

The screenshot displays the dCode Cipher Identifier website interface. At the top, there's a banner for a dCode Survey. The main heading is "CIPHER IDENTIFIER" with a sub-header "Cryptography > Cipher Identifier". Below this, a search bar contains the text "e.g. type 'caesar'". The results section, titled "Results", shows "dCode's analyzer suggests to investigate:" followed by a list of cipher types: Atbash Cipher, ROT Cipher, Caesar Cipher, Substitution Cipher, Homophonic Cipher, Chaocipher, and Mono-alphabetic Substitution. The "Atbash Cipher" is highlighted. To the right, there's a section for "CODING MESSAGE IDENTIFIER" with a "CIPHERTEXT TO RECOGNIZE" field containing a sample ciphertext. Below this is an "ANALYZE" button. Further down, there's a link to "See also: Frequency Analysis — Index of Coincidence" and a "SYMBOLS IDENTIFIER" section with a "Go to: Symbols Cipher List" link. On the far right, a "Summary" sidebar lists various features and FAQs, and a "Similar pages" section lists related tools like Frequency Analysis, Index of Coincidence, and Symbols Cipher List.

# Traficant

În acest challenge avem o captura de date.

Deschizand wireshark putem observa trafic transmis HTTP.

Accesand: *File>Export Objects>HTTP...* putem observa un fișier mai ciudat.(poza pe slide-ul următor)

După ce am descărcat fișierul, vom găsi flag-ul encodat in Base64

[https://gchq.github.io/CyberChef/#recipe=From\\_Base64\('A-Za-z0-9%2B/%3D',true\)&input=VWxOVWV6RTVaamt5WldObVltRTBOMkl3WVRObU56aGhNak5rTnpNMVIUSmhZekJqTnpobFIXTTNORGRtTUdRNE5UTTJORE16T1RWbVIURmINRE0zWXpaa1pqUjk](https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)&input=VWxOVWV6RTVaamt5WldObVltRTBOMkl3WVRObU56aGhNak5rTnpNMVIUSmhZekJqTnpobFIXTTNORGRtTUdRNE5UTTJORE16T1RWbVIURmINRE0zWXpaa1pqUjk)

Flag: **RST{19f92ecfba47b0a3f78a23d735a2ac0c78eac747f0d853643395fa1b037c6df4}**

Wireshark - Export - HTTP object list

Text Filter: Content Type: All Content-Types

Packet	Hostname	Content Type	Size
6032	1153288396.rsc.cdn77.org	image/png	3,087 byte
6034	1153288396.rsc.cdn77.org	image/png	3,104 byte
6036	ocsp.digicert.com	application/ocsp-response	471 bytes
6038	1153288396.rsc.cdn77.org	image/png	2,746 byte
6040	1153288396.rsc.cdn77.org	image/png	2,872 byte
6054	www.http2demo.io	image/x-icon	1,150 bytes
7000	detectportal.firefox.com	text/plain	8 bytes
7156	dns-asta-poate-fi-util-atunci-cand-merge.rstcon.com	text/plain	149 bytes
7753	ocsp.pki.goog	application/ocsp-request	83 bytes
7755	ocsp.pki.goog	application/ocsp-response	471 bytes
9168	detectportal.firefox.com	text/plain	8 bytes
9256	ocsp.digicert.com	application/ocsp-request	83 bytes
9258	ocsp.digicert.com	application/ocsp-response	471 bytes
9308	ocsp.digicert.com	application/ocsp-request	83 bytes
9313	ocsp.digicert.com	application/ocsp-response	471 bytes
9315	ocsp.digicert.com	application/ocsp-request	83 bytes
9325	ocsp.digicert.com	application/ocsp-request	83 bytes
9329	ocsp.digicert.com	application/ocsp-request	83 bytes

Frame 7156: 561 bytes on wire (4488 bits) captured (561 bytes) over Ethernet (Ethernet II) ...  
Linux cooked capture v1  
Internet Protocol Version 4, Src: 104.046109863, Dst: 127.0.0.1  
Transmission Control Protocol, Src Port: 3518, Dst Port: 127.0.0.1  
Hypertext Transfer Protocol  
Line-based text data: text/plain

0000 00 00 03 04 00 06 00  
0010 45 00 02 21 36 4e 40  
0020 7f 00 00 01 00 50 ea  
0030 80 18 02 00 00 16 00  
0040 98 31 33 bf 48 54 54  
0050 20 4f 4b 0d 0a 44 61  
0060 31 34 20 41 70 72 20 32 30 32 31 20 32 31 3a 34 14 Apr 2 021 21:4  
0070 39 3a 33 39 20 47 4d 54 0d 0a 53 65 72 76 65 72 9:39 GMT .Server  
0080 3a 20 41 70 61 63 68 65 2f 32 2e 34 2e 34 36 20 : Apache /2.4.46  
0090 28 44 65 62 69 61 6e 29 0d 0a 4c 61 73 74 2d 4d (Debian) .Last-M  
00a0 6f 64 69 66 69 65 64 3a 20 57 65 64 2c 20 31 34 odified: Wed, 14

Frame (561 bytes) Uncompressed entity body (149 bytes)

Captura.pcapng Packets: 11307 · Displayed: 11307 (100.0%) Profile: Default

# Crackpass

Pentru acesta, am folosit ghidra, dar înainte am rulat comanda **strings**.

Am descoperit un string interesant :

[illegible]

În ghidra am putut observa ca se efectuează o operație de XOR.(Page 8)

Urmărim funcția “enc” vom afla cheia folosită pt XOR.(Page 9)

[illegible]

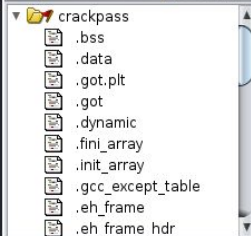
Flag : RST{93732ff5e18683740582e443e0cc4f6f8201fb9eebf1204537eb05c53ca5be9d}



File Edit Analysis Graph Navigation Search Select Tools Window Help



Program Trees



Program Tree ×

Symbol Tree

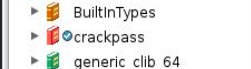


Filter:

Data Type Manager



Data Types



Listing: crackpass - (7 addresses selected)

00101240	48 89 c7	MOV	RDI, RAX
00101243	e8 18 fe ff ff	CALL	length
00101248	48 39 45 e8	CMP	qword ptr [RBP + local_20], RAX
0010124c	0f 92 c0	SETC	AL
0010124f	84 c0	TEST	AL, AL
00101251	74 62	JZ	LAB_001012b5
00101253	48 8d 15	LEA	RDX, [enc]
	26 2e 00 00		
0010125a	48 8b 45 e8	MOV	RAX, qword ptr [RBP + local_20]
0010125e	48 01 d0	ADD	RAX, RDX
00101261	0f b6 18	MOVZX	EBX, byte ptr [RAX] => enc
00101264	48 8b 55 e8	MOV	RDX, qword ptr [RBP + local_20]
00101268	48 8d 45 c0	LEA	RAX => local_48, [RBP + -0x40]
0010126c	48 89 d6	MOV	RSI, RDX
0010126f	48 89 c7	MOV	RDI, RAX
00101272	e8 29 fe ff ff	CALL	operator[]
	ff ff		
00101277	0f b6 00	MOVZX	EAX, byte ptr [RAX]
0010127a	31 c3	XOR	EBX, EAX
0010127c	48 8b 45 e8	MOV	RAX, qword ptr [RBP + local_20]
00101280	48 89 c6	MOV	RSI, RAX
00101283	48 8d 3d	LEA	RDI, [none[abi:cxx11]]
	76 2e 00 00		
0010128a	e8 11 fe ff ff	CALL	operator[]
	ff ff		
0010128f	0f b6 00	MOVZX	EAX, byte ptr [RAX]
00101292	38 c3	CMP	BL, AL
00101294	0f 95 c0	SETNZ	AL
00101297	84 c0	TEST	AL, AL
00101299	74 13	JZ	LAB_001012ae
0010129b	48 8d 3d	LEA	RDI, [s_Steagul_este_gresit!_0010203f]
	9d 0d 00 00		
001012a2	e8 c9 fd ff ff	CALL	puts
	ff ff		
001012a7	bb 03 00 00 00	MOV	EBX, 0x3
001012ac	eb 18	JMP	LAB_001012c6

Decompile: main - (crackpass)

```

long local_20;

f (param_1 == 2) {
    std::allocator<char>::allocator();
    /* try { // try from 001011f6 to 001011fa has its CatchHandler @ 0
    std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::bas
        ((char *)local_48, *(allocator **) (param_2 + 8));
    std::allocator<char>::~allocator(&local_21);
    lVar3 = std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>
    if (lVar3 == 0x45) {
        local_20 = 0;
        while( true ) {
            uVar4 = std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<
                ();
            if (uVar4 <= local_20) break;
            bVar1 = enc[local_20];
            pbVar5 = (byte *)std::__cxx11::
                basic_string<char, std::char_traits<char>, std::allocator<char>
                    ((ulong)local_48);

            bVar2 = *pbVar5;
            pbVar5 = (byte *)std::__cxx11::
                basic_string<char, std::char_traits<char>, std::allocator<char>
                    ((ulong)none[abi:cxx11]);

            if ((bVar1 ^ bVar2) != *pbVar5) {
                puts("Steagul este gresit!");
                uVar6 = 3;
                goto LAB_001012c6;
            }
            local_20 = local_20 + 1;
        }
        puts("Da, acela e steagul!");
        uVar6 = 0;
    }
    else {
        /* try { // try from 00101225 to 001012c0 has its CatchHandler @ 0
        puts("Steagul are 69 de caractere!");
        uVar6 = 2;
    }
}

```

Console - Scripting

\_ □ ×



```

uVar4 = std::__cxx11::basic_string<char,std::char_traits<
    >
    >();
if (uVar4 <= local_20) break;
bVar1 = enc[local_20];
pbVar5 = (byte *)std::__cxx11::
    basic_string<char,std::char_traits<char>
    >((ulong)local_48);
bVar2 = *pbVar5;

```

.....	.. .. .. .
04060	00 00 00 00 00 00 00 00 68 40 10 00 00 00 00 00
04070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04080	02 1f 19 00 08 02 06 02 03 57 57 04 54 00 09 07
04090	09 02 06 05 01 04 09 03 54 05 05 02 54 01 52 52
040a0	05 57 07 57 09 03 01 00 57 53 08 54 54 53 57 00
040b0	03 01 05 04 02 06 54 53 01 04 52 04 02 52 50 04
040c0	53 54 08 55 00 00 00 00 28 50 10 00 00 00 00 00
.....	.. .. .. .
040e0	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??

# Hecagon

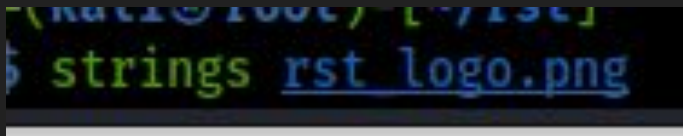
In poza atasata putem recupera flag-ul folosind comanda **strings**.

```
strings vacanta la oache.jpg
```

Flag : RST{1jd90dsnuaicnavdx0vidj91szmxsnvbedas}

# Sigla

Abordarea chall-ului este identică cu rezolvarea chall-ului precedent.



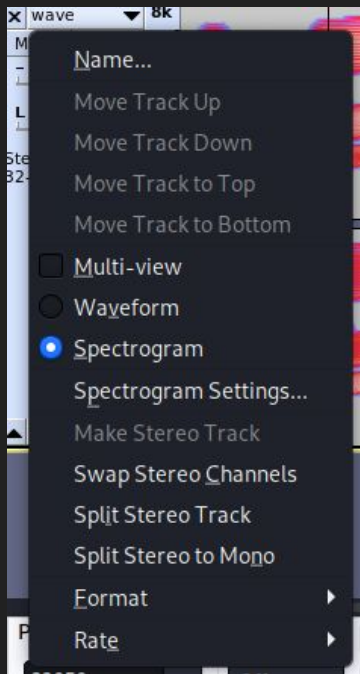
```
(kali@root) [~/13C]  
$ strings rst_logo.png
```

Flag:

RST{a2e6d2358601328bfc06c9c9a42a4f644ed6303fd1b2d696eb4d438c5c1a7144}

# Valuri

Avem un fișier .wav. L-am deschis în **audacity** si am aplicat filtrul **Spectrogram**.



În acest mod, am descoperit mesajul.



Aplicam MD5 [https://gchq.github.io/CyberChef/#recipe=MD5\(\)&input=RjUyTFIHNFo](https://gchq.github.io/CyberChef/#recipe=MD5()&input=RjUyTFIHNFo)

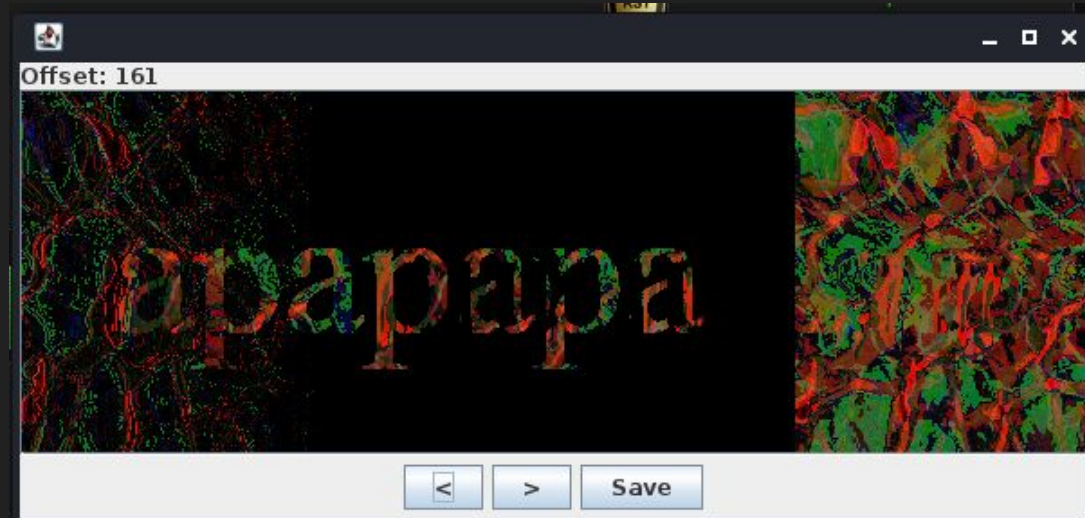
Flag : **RST{ec627195bc62796ae3471b7a1c28d059}**

# Stegano II

Pentru acest chall am folosit **stegsolve**.

Folosind stereograma(Analyse>Stereogram Solver),ajustam offsetul astfel încât mesajul să poată fi citibil

Mesaj: “apapapa”



# Stegano II

Aplicăm SHA256

[https://gchq.github.io/CyberChef/#recipe=SHA2\('256',64,160\)&input=YXBhcGFwYQ](https://gchq.github.io/CyberChef/#recipe=SHA2('256',64,160)&input=YXBhcGFwYQ)

Flag :

**RST{aea56887a4f67f2b7937a7965fa679bfae9358695ebbd7f28470c9ba436fd93c}**

# Elevi

Am intrat pe site și am inspectat fiecare pagina(F12).

Pagina “Despre” pare sa aibă un comentariu interesant

```
▶ <p>...</p>
▶ <h4>...</h4>
  <p style="text-align:left;">Lista cu administratorii paginii</p>
▶ <ul>...</ul>
▶ <style>...</style>
  <!--Mihai, nu uita sa verifici fisierele css! (http://127.0.0.1/shell.php) -ANDRADA--->
▶ <div class="footer">...</div>
  <p></p>
</body>
</html>
```

Am accesat fișierul “shell.php” și am observat ca pot sa rulez comenzi pe server.

Inițial am încercat sa fac un reverse shell care sa se conecteaza la sistemul meu,dar fără succes.



# Elevi

Ruland comanda “ls”, observam un fișier “readme” în care se afla următorul text:

**Result of command execution:**

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

201dc88daf95daf8a18ae57d0691d547ac3a5ce77d15620ec6b109eef9697917f7e18ada1d2c7620ab17d9b3e476197e

După câteva încercări eșuate de a trece prin fișierele din /etc,am descoperit o cheie în “/etc/hosts”

#key 7B7B526F6D616E69616E53656375726974795465616D7D7D

Probabil ca cipher text-ul este cel menționat din “readme”

# Elevi

Algoritmul folosit este **AES**

[https://gchq.github.io/CyberChef/#recipe=AES\\_Decrypt\(%7B'option': 'Hex', 'string': '7B7B526F6D616E69616E53656375726974795465616D7D7D%20'%7D,%7B'option': 'Hex', 'string': '7B7B526F6D616E69616E53656375726974795465616D7D7D%20'%7D,'CBC','Hex','Raw',%7B'option': 'Hex', 'string': '%7D,%7B'option': 'Hex', 'string': '%7D'\)&input=MjAxZGM4OGRhZjk1ZGFmOGExOGFINTdkMDY5MWQ1NDdhYzNhNWNINzdkMTU2MjBIYzZiMTA5ZWVmOTY5NzIxN2Y3ZTE4YWRhMWQyYzc2MjBhYjE3ZDliM2U0NzYxOTdlIA](https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option': 'Hex', 'string': '7B7B526F6D616E69616E53656375726974795465616D7D7D%20'%7D,%7B'option': 'Hex', 'string': '7B7B526F6D616E69616E53656375726974795465616D7D7D%20'%7D,'CBC','Hex','Raw',%7B'option': 'Hex', 'string': '%7D,%7B'option': 'Hex', 'string': '%7D')&input=MjAxZGM4OGRhZjk1ZGFmOGExOGFINTdkMDY5MWQ1NDdhYzNhNWNINzdkMTU2MjBIYzZiMTA5ZWVmOTY5NzIxN2Y3ZTE4YWRhMWQyYzc2MjBhYjE3ZDliM2U0NzYxOTdlIA)

Flag : **RST{sacmxke0xakosoicd31jsqd01lsaly10san2}**