
GCSC 2022

Leox



2022-02-24

Madagascar

Flag : GCSC2022{https://discord.gg/3gvtd8N6}

Le lien était présent dans le mail reçu contenant les informations de connexion

Kenya

- pseudonyme : CyberBadCorp
- email : cyberbadcorp@gmail.com
- token d'inscription : 20022022

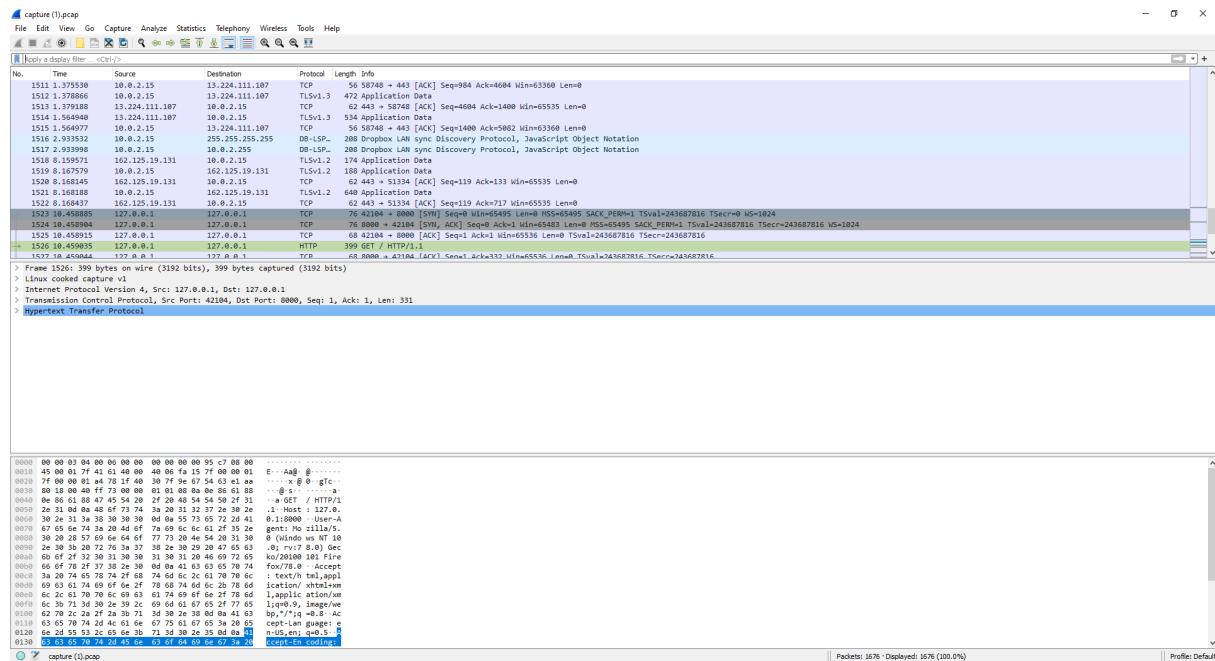
Flag : GCSC2022{mot_de_passe}

Il est possible de reconstruire le mot de passe en suivant le format de celui que l'on a reçu par mail : GCSC2022_PseudoTokenEmailToken

Flag : GCSC2022{GCSC2022_CyberBadCorp20022022cyberbadcorp@gmail.com20022022}

Somalia

J'ai ouvert la capture et l'ai ouverte dans Wireshark :



J'ai vu qu'il y avait des paquets HTTP, j'ai donc commencé par là

```
GET / HTTP/1.1
Host: 127.0.0.1:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.9.2
Date: Fri, 04 Feb 2022 19:23:06 GMT
Content-type: text/html; charset=utf-8
Content-Length: 344

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="paquets.jpg">paquets.jpg</a></li>
</ul>
<hr>
</body>
</html>
```

1 client pkt 1 server pkt 1 turn.

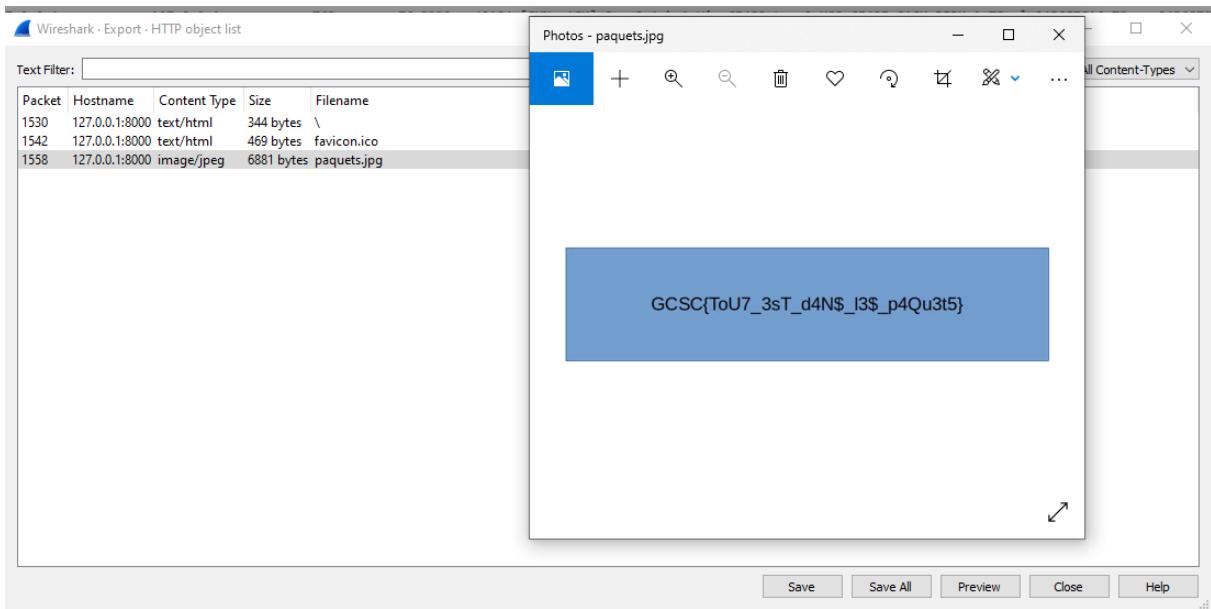
Entire conversation (829 bytes)

Show data as ASCII

Find: Find Next

Ici, on voit qu'il y a [paquets.jpg](#) dans le paquet, ce qui veut dire qu'on pourrait peut-être récupérer cette image en allant dans : File -> Export Objects -> HTTP (HTTP car paquet HTTP)

Et effectivement on voit l'image, on peut donc la télécharger



Flag : GCSC2022{ToU7_3sT_d4N\$_l3\$p4Qu3t5}

South Africa

J'ai téléchargé l'image, l'ai ouvert dans un navigateur et ai regardé le code source (étant donné que c'est un svg)



Dans le code, on peut voir qu'il semble y avoir une seconde "couche" :

```

17058 W5WdZbZVNTBkvA+enHO4RVmwNUkkNy7nAFznAxBZw2wowzyyGhODt4UFwD4EUWEilaiSCKFvGwV4
17059 /vzw1u07Jyen49G0qCoisIVdxV27PnixUuXvjc2YoLzysKRg2aRQEBIAaq1k2R55ATzpcvczISQ
17060 Ku+ImBtf8tIMGQprxcfSFcTE1rjCxeCbpq6GVtUcLNq2D4Gcda4YT6eDwTck9P8A3ZkAscYjcDwA
17061 AAAASUVORK5CYII=
17062 "
17063     id="image17"
17064     x="-4.3955674"
17065     y="24.974348" />
17066     <image
17067     width="12.062613"
17068     height="1.5402763"
17069     preserveAspectRatio="none"
17070     xlink:href="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAArkAAABZCAYAAA6hTwfAAAABHNCsvQICAgIfAhkiAAAAnxJREFU
eJzt3EFuwjAUQMGm6v2v7K5YFMytC1gpldk+D8Khg9seAYY4wvAAAI+x73AAAA8GwiFwCAHJEL
17071 AECOyAUAIefkAgCQ8zM7cRzHn79vP8Jw037/owyz41fxPbv+s87P51odn51/97yr+XaPv+q53x+f
17072 vx73frPrln2uq/l35929bnWfs/Pdv/7rdVfzzu7/6H1x/8fZ0xb32wqe/34/707z3X30qfv10c/j
17073 2bzq6161b67uo9lcn/bcVsdrn5/N05vj6n07ut5q3tX53f242xtn5919P8zWm12/0+eVzzPf5AIA
17074 kCMyAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05IXAAcKQu
17075 AA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBAjsgFACBH
17076 5AIakCNYAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05IXAA
17077 ckQuAA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBAjsgF
17078 ACBH5AIakCNYAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05I
17079 XAAcKQuAA5IhcAgByRCwBAjsgFACBH5AIakCNYAQDIebkAA05IXAAcKQuAA5IhcAgByRCwBA
17080 jsgFACBH5AIakCNYAQDIebkAA0QcY4zx7iEAAOCzfJMLAECoyAUAIefkAgCQI3IBAMgRuQAA5Ihc
17081 AAByfgFRxiK6XYCKsAAAAABJRUErkJggg=
17082 "
17083     id="image29"
17084     x="150.91969"
17085     y="70.401352"
17086     transform="rotate(38.524062)" />
17087     <image
17088     width="12.062613"
17089     height="1.5402763"

```

J'ai donc copié cette string base64 et l'ai collé dans un site permettant de le transformer en image (<https://codebeautify.org/base64-to-image-converter>) :

The screenshot shows a web application titled "Base64 to Image". On the left, there is a text input field labeled "Enter Base64 String" containing a long base64 encoded string. On the right, there are several buttons: "Sample" (disabled), "Auto Update" (checked), "Generate Image" (highlighted in green), "File" (disabled), "URL" (disabled), and "Download Image". Below the input field, it says "Size : 993 B, 982 chars". In the center, there is a large barcode representation of the input string. At the bottom, there is a blue "Download Image" button.

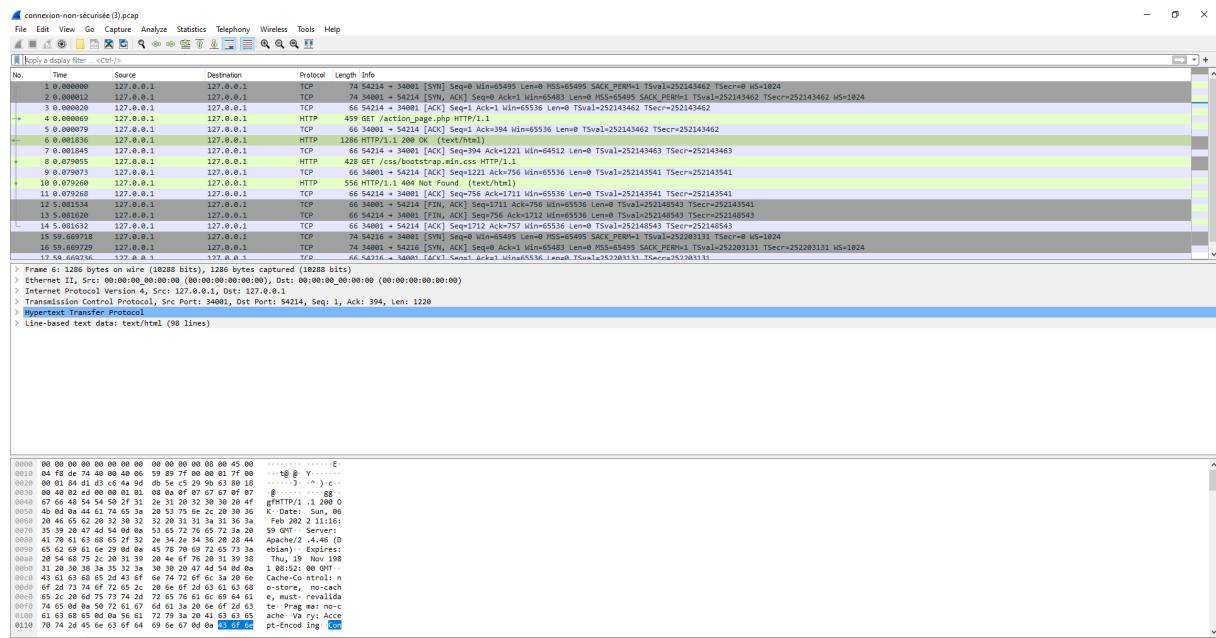
Maintenant, on peut essayer de lire le code bar : (<https://products.aspose.app/barcode/fr/recognize#>) :

The screenshot shows a web application titled "Lecteur de codes barres en ligne". It has a blue header with the title and a message: "Téléchargez votre image, choisissez le type de code-barres ou laissez « Tous les types » et cliquez sur le bouton « Lire le code à barres ». Propulsé par [aspose.com](#) et [aspose.cloud](#)". Below this is a green "Une autre image" button. In the center, there is a barcode image. To its left, it says "Type: Code128". To its right, there is a text input field containing "GCSC{baR_c0d3\$_oR_J4lL_b4rZ}" with a "Générer un nouveau" button next to it. At the bottom, there is a green "Modifier les paramètres de reconnaissance" button.

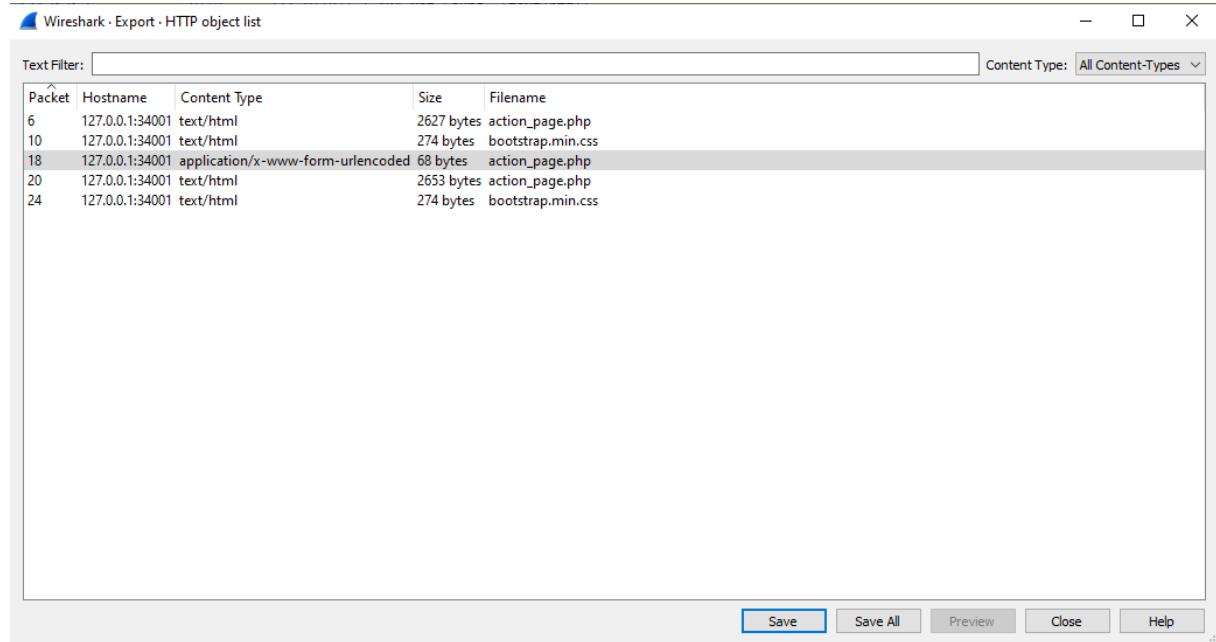
Flag : GCSC2022{baR_c0d3\$_oR_J4lL_b4rZ}

Zambia

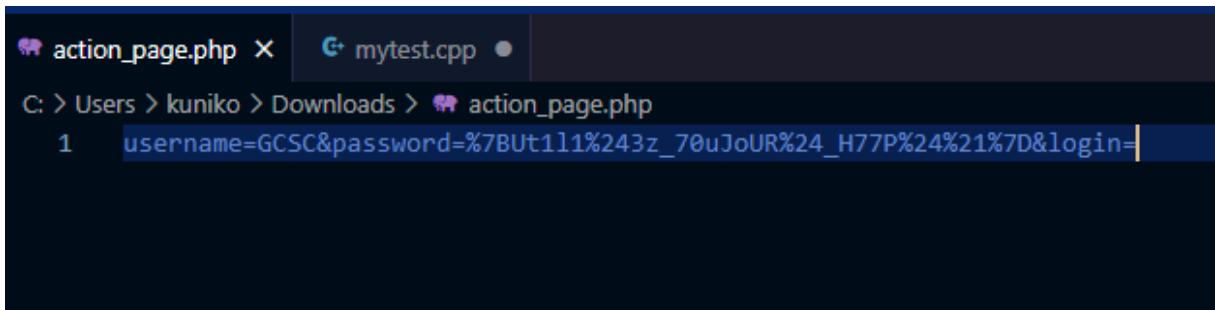
J'ai téléchargé la capture réseau et l'ai mise dans Wireshark :



Je n'ai rien trouvé de spécial ici :(, donc j'ai regardé une fois de plus dans File -> Export Objects -> HTTP, et là j'y ai trouvé quelque truc :

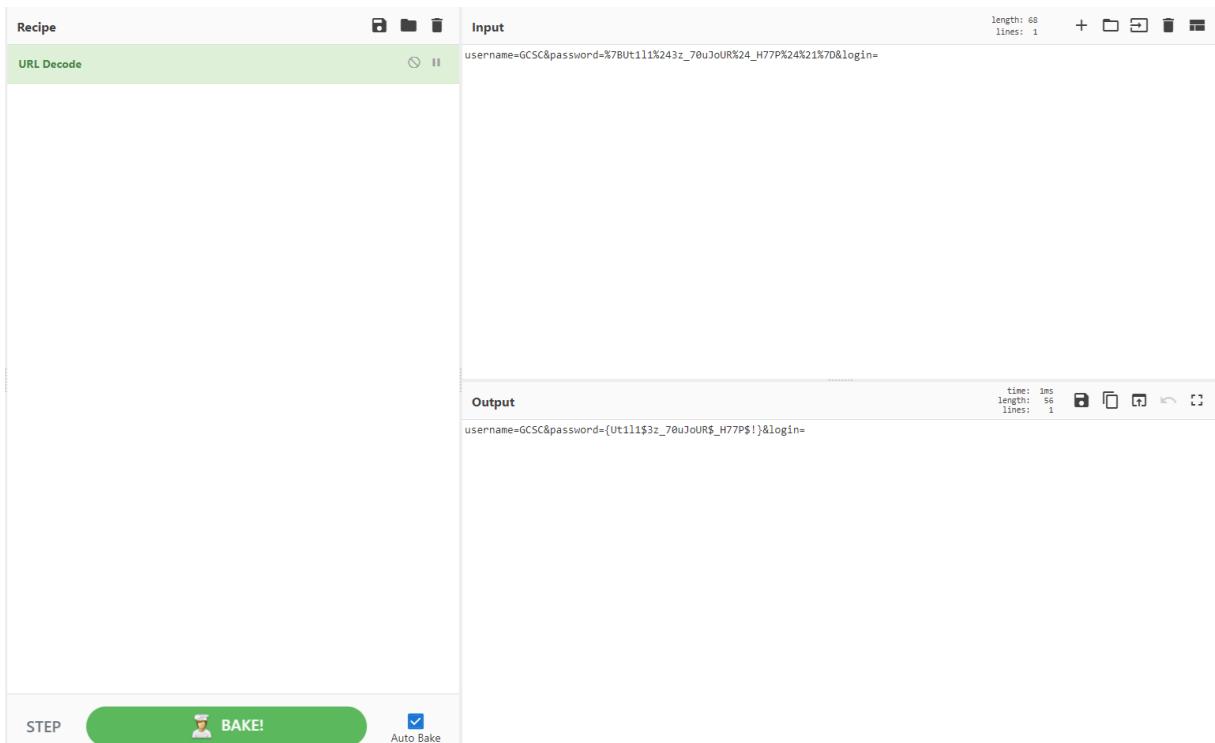


J'ai téléchargé le `action_page.php` avec le Content Type `application/x-www-form-urlencoded` et l'ai ouvert, j'ai trouvé :



The screenshot shows a terminal window with two tabs: "action_page.php" and "mytest.cpp". The "action_page.php" tab is active, displaying the command "C: > Users > kuniko > Downloads > action_page.php" and the output of a command that has been redacted.

J'ai collé ça dans CyberChef (<https://gchq.github.io/CyberChef/>) :

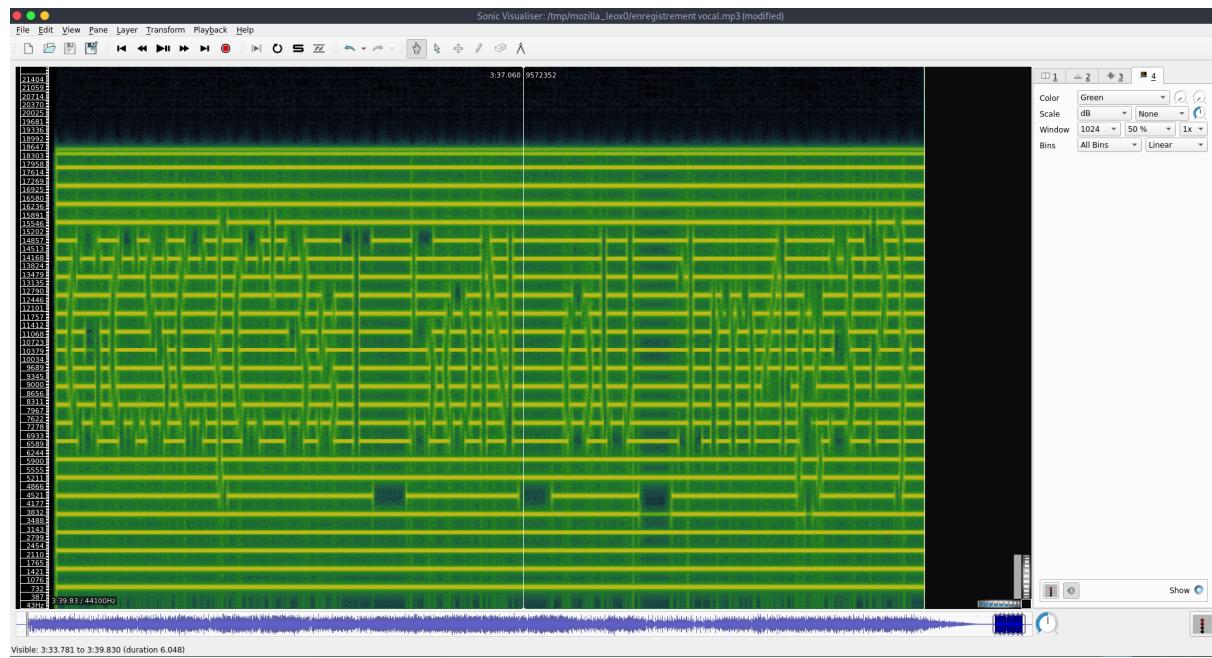


The screenshot shows the CyberChef interface. The "Input" panel contains the URL: "username=GCSC&password=%78Ut1l1%243z_70uJoUR%24_H77P%24%21%7D&login=". The "Output" panel shows the decoded version: "username=GCSC&password={Ut1l1\$3z_70uJoUR\$_H77P\$!}&login=". The "URL Decode" step is highlighted in green at the bottom of the recipe list.

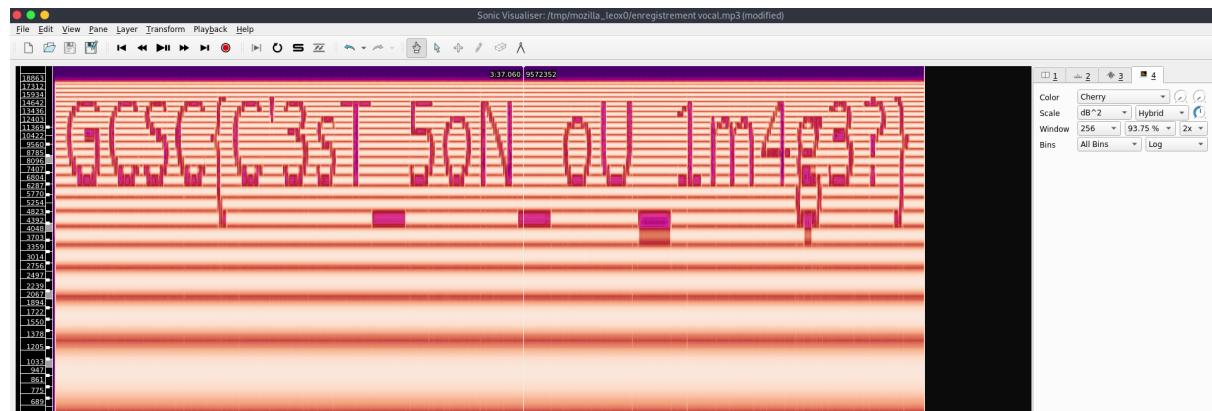
Flag : GCSC2022{Ut1l1\$3z_70uJoUR\$_H77P\$!}

Angola

J'ai téléchargé le .mp3 (je me suis fait rickroll au passage) puis je l'ai mis dans Sonic Visualizer et ai ajouté un effect Spectograms pour essayer de trouver du texte caché :



Il y'a effectivement quelque chose, j'ai donc appliqué quelque filtre pour mieux y voir :



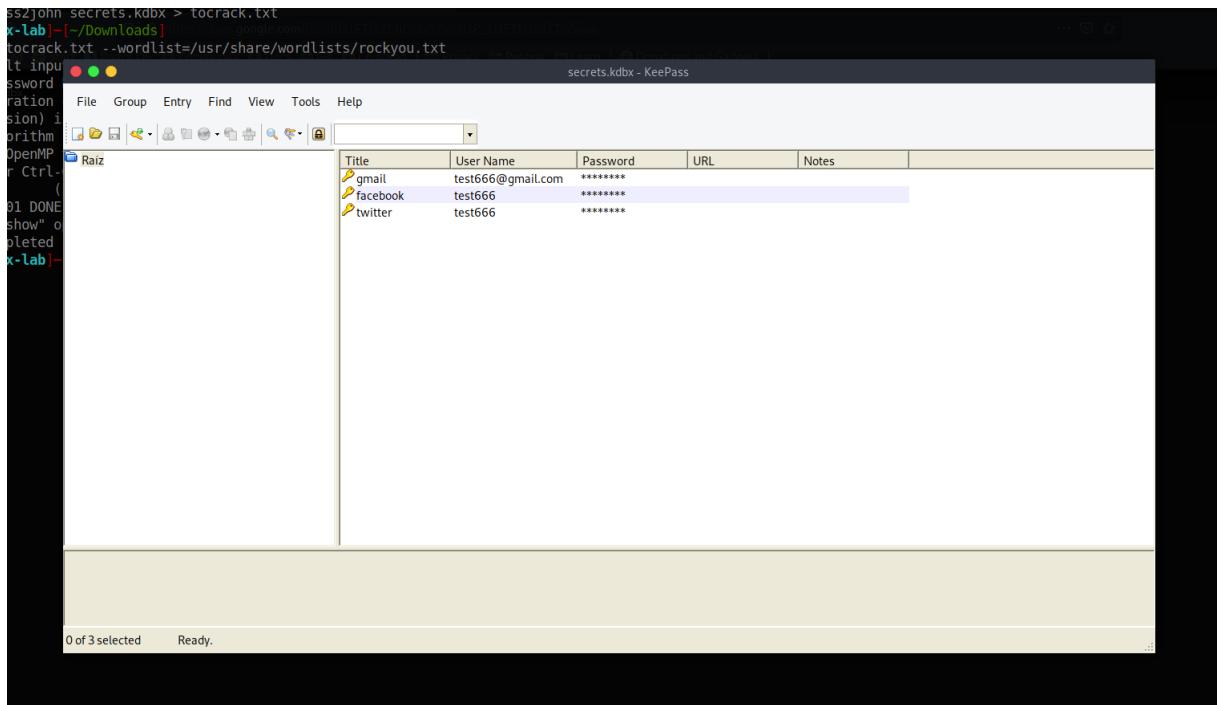
Flag : GCSC2022{C'3sT_5oN_oU_1m4g3?}

Congo - Kinshasa

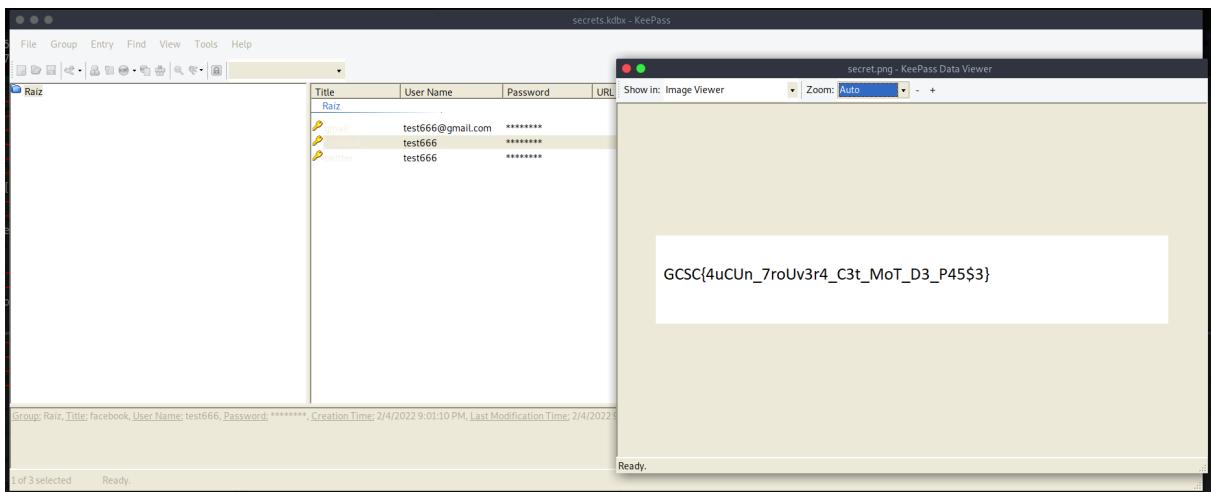
Le fichier semble être une db de mot de passe Keepass, on peut essayer de cracker ça :

```
[leox@leox-lab]~/Downloads] $ keepass2john secrets.kdbx > tocrack.txt
[leox@leox-lab]~/Downloads] https://drive.google.com/file/d/1zIIFTxZMIDr9xyFVMtsP_OlF7M1Gt2Tn/view
[leox@leox-lab]~/Downloads] $ john tocrack.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 30 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES, 1=TwoFish, 2=ChaCha]) is 0 for all loaded hashes
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
rellik000      (secrets)
1g 0:00:00:01 DONE (2022-02-20 18:26) 0.5025g/s 408042p/s 408042c/s 408042C/s rellik000..reller
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[leox@leox-lab]~/Downloads]
[leox@leox-lab]~/Downloads]
```

Le mot de passe de la DB semble être “rellik000”, on peut donc maintenant ouvrir la DB :



On peut voir qu'il y'a une photo attaché au pass de facebook :



Flag : GCSC2022{4uCUn_7roUv3r4_C3t_MoT_D3_P45\$3}

Chad

L'archive .zip est protégé par mot de passe, on peut essayer de cracker ça :

```
[leox@leox-lab] -[~/Downloads] $ unzip note.zip
Archive: note.zip
[note.zip] note.txt password:
  skipping: note.txt      incorrect password
[x]-[leox@leox-lab] -[~/Downloads] $ zip2john note.zip > tocrack.txt
ver 2.0 note.zip/note.txt PKZIP Encr: cmplen=47, decmplen=35, crc=D4124615
[leox@leox-lab] -[~/Downloads] $ john tocrack.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Catsandcows      (note.zip/note.txt)
1g 0:00:00:01 DONE (2022-02-20 18:47) 0.7575g/s 8559Kp/s 8559Kc/s 8559KC/s ChEYENNE..Cambridge07
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[leox@leox-lab] -[~/Downloads] $
```

On peut donc maintenant unzip l'archive :

```
[leo@leo-lab]~/Downloads]
└─$ unzip note.zip
Archive: note.zip
[note.zip] note.txt password:
extracting: note.txt
[leo@leo-lab]~/Downloads]
└─$ cat note.
cat: note.: No such file or directory
[x] [leo@leo-lab]~/Downloads]
└─$ cat note.txt
GCSC{p4$sw0rD_cr4ck1nG_br34kZ_GPUs} [leo@leo-lab]~/Downloads]
└─$
```

The terminal session shows the extraction of a ZIP file named 'note.zip' from the current directory. The file contains a single text file named 'note.txt'. The terminal then attempts to read the contents of 'note.' but fails because it does not exist. Finally, the contents of 'note.txt' are displayed.

Flag : GCSC2022{p4\$sw0rD_cr4ck1nG_br34kZ_GPUs}

Nigeria

C'est un fichier .db, on peut donc essayer de l'ouvrir

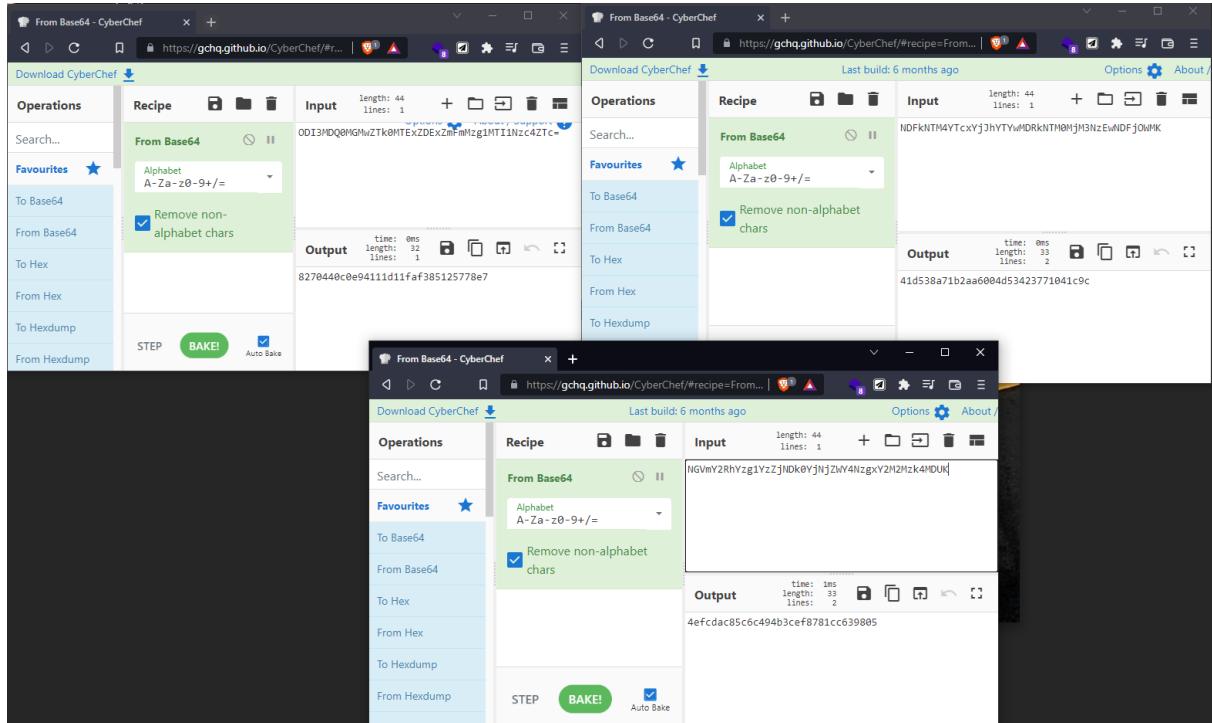
On peut voir les mots de passe :

The screenshot shows the DB Browser for SQLite interface. A SQL query 'SELECT * FROM users;' is run, and the results are displayed in a table:

username	password
John	ODI3MDQ0MGWzTk0MTExZDExZmFmMz...
Marie	NDFkNTM4YTcxYjhYTtywMDRkNTM0MjM3N...
Luc	NGVmY2RhYzg1YzZjNDk0YjNjZWY4NzgxY...

Execution finished without errors.
Result: 3 rows returned in 5ms
At line 1:
SELECT * FROM users;

Ils sont en base64, on va donc “reverse” ça :



Maintenant, ça semble être du md5, on va mettre ces trois hash dans un fichier texte et utiliser john.

```
[leox@leox-lab] -[~/Downloads]
└─$ cat md5.txt
8270440c0e94111d11faf385125778e7
41d538a71b2aa6004d53423771041c9c
4efcdac85c6c494b3cef8781cc639805

[leox@leox-lab] -[~/Downloads]
└─$ john md5.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=3
Press 'q' or Ctrl-C to abort, almost any other key for status
p1ntap      (?)
fd1972      (?)
AdDiCtIoN   (?)
[leox@leox-lab] -[~/Downloads]
└─$
```

Flag : GCSC2022{p1ntap_fd1972_AdDiCtIoN}

Libya

L'intitulé du chall nous donne un indice, le message utilise les chiffres des templiers :



On peut donc décoder ça :

dCode

Rechercher un outil

- RECHERCHE SUR DCODE PAR MOTS-CLÉS : Tapez par exemple 'sudoku'
- PARCOURIR LA LISTE COMPLÈTE DES OUTILS

Résultats

GCSCSECRETOFTEMPLAIRES

Chiffre des Templiers - [dCode](#)

Catégorie(s) : Substitution par Symboles

Partager

+ f t g m

dCode et plus

dCode est gratuit et ses outils sont une aide précieuse dans les jeux, les maths, les énigmes, les géocaches.

CHIFFRE DES TEMPLIERS

Cryptographie > Chiffrement par Substitution > Substitution par Symboles > Chiffre des Templiers

DÉCHIFFREMENT DU CODE TEMPLIERS

SYMBOLES DE L'ALPHABET TEMPLIERS (CLIQUER POUR AJOUTER)

<	>	>>	<<	<<<	>>>	<<<<	>>>>	<<<<<	>>>>>
>	<	<>	><	<<>	>><	<<<>	>>><	<<<<>	>>>><
>>	<<	<<<	>>>	<<<<	>>>>	<<<<<	>>>>>	<<<<<<	>>>>>>
<<<	>>>	<<<<	>>>>	<<<<<	>>>>>	<<<<<<	>>>>>>	<<<<<<<	>>>>>>>

MESSAGE CHIFFRÉ PAR CODE DES TEMPLIERS

<	>	>>	<<	<<<	>>>	<<<<	>>>>	<<<<<	>>>>>
>	<	<>	><	<<>	>><	<<<>	>>><	<<<<>	>>>><
>>	<<	<<<	>>>	<<<<	>>>>	<<<<<	>>>>>	<<<<<<	>>>>>>
<<<	>>>	<<<<	>>>>	<<<<<	>>>>>	<<<<<<	>>>>>>	<<<<<<<	>>>>>>>

DÉCHIFFRER

Voir aussi : Chiffre Pig Pen des Francs-maçons

Menu

- * Déchiffrement du Code Templiers
- * Chiffrement avec Code Templiers
- * Qu'est ce que le chiffre des Templiers ? (Définition)
- * Comment encoder avec le chiffre des Templiers ? (Principe de chiffrement)
- * Comment décoder avec le Chiffre des Templiers ? (Principe de déchiffrement)
- * Comment reconnaître le chiffre des Templiers ?
- * Pourquoi manque-t-il la lettre J ?
- * Qui a créé le Chiffre des Templiers ?

Flag : GCSC2022{secret_of_templairs}

Algeria

J'ai remplacé tout les "Dash" par – et tout les "Dot" par ., ce qui donne :

```
1
2 -... .-.-. ....
3 -... .-.-. ....
4 -...
5 -...
6 -... .-.-. ....
7 -...
8 -...
9 -...
10 -...
11 -...
12 -...
13 -...
14 -...
```

C'est du morse, on peut donc convertir ça :

length: 410
lines: 14

From Morse Code

Letter delimiter: Space Word delimiter: Line feed

Input

```
...- .-.-. ....
...- .-.-. ....
-...
-...
-... .-.-. ....
-...
-...
-...
-...
-...
-...
-...
-...
```

Output

```
time: 2ms
length: 108
lines: 1
```

CHERS COLLEGUES DE LA CYBERBADCORP, LANCEZ L'OPRATION "REMORELESS", LE SAMEDI 12/02/2022 23:59 @GCSTF.TEAM

Flag : GCSC2022{remorseless}

Russia

Il est écrit “On est d'accord que la méthode des hackers n'a rien d'ordinaire”, le mot “méthode” est un indice.

On peut lister la liste des méthodes HTTP autorisé par le serveur avec la méthode “OPTIONS” :

```
[leox@leox-lab] -[~/Downloads]
└─$ curl -X OPTIONS -s http://challenges.guinean-cybertaskforce.com:8001/ -v
* Trying 159.65.242.247:8001...
* Connected to challenges.guinean-cybertaskforce.com (159.65.242.247) port 8001 (#0)
> OPTIONS / HTTP/1.1
> Host: challenges.guinean-cybertaskforce.com:8001
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: text/html; charset=utf-8
< Allow: OPTIONS, GCSC
< Content-Length: 0
< Server: Werkzeug/2.0.3 Python/3.8.10
< Date: Sun, 20 Feb 2022 18:38:52 GMT
<
* Closing connection 0
[leox@leox-lab] -[~/Downloads]
└─$ 
```

capture_Russ
la méthode d'ordinaire?

La CyberBadCorp s'est d'Information à la su
Ils ont pris de nouve
sécurité.

[Link 1]

La méthode “GCSC” n'est effectivement pas ordinaire.

J'ai essayé plusieurs trucs avec, mais rien ne marchais, jusqu'à ce que j'essaie de reach /flag :

```
[leox@leox-lab] -[~/Downloads]
└─$ curl -X GCSC -s http://challenges.guinean-cybertaskforce.com:8001/
Qu'est ce que c'est ce methode?? [leox@leox-lab] -[~/Downloads]
└─$ curl -X GCSC -s http://challenges.guinean-cybertaskforce.com:8001/flag
GCSC{p0s7_Et_g3t_ne_$ont_p4$_sp3c13lle5} [leox@leox-lab] -[~/Downloads]
└─$ 
```

Flag : GCSC{p0s7_Et_g3t_ne_\$ont_p4\$_sp3c13lle5}

Canada

Ici, il y'a juste un formulaire basique :

Please, register a new user to continue

user
password
email
test.com
Submit

J'ai essayé d'ajouter une quote au site web personnel pour voir si un message d'erreur allais être retourné et c'est le cas :

requests.exceptions.ConnectionError

```
requests.exceptions.ConnectionError: HTTPConnectionPool(host='hello.com32', port=80): Max retries exceeded with url: /GCSC%7BUtillisseZ_t0ujoUr$_c0ll4boRat0R%7D (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f74c0889310>: Failed to establish a new connection: [Errno -2] Name or service not known'))
```

Traceback (most recent call last)

```
File "/usr/local/lib/python3.8/dist-packages/urllib3/connection.py", line 174, in _new_conn
    conn = connection.create_connection()
File "/usr/local/lib/python3.8/dist-packages/urllib3/util/connection.py", line 72, in create_connection
    for res in socket.getaddrinfo(host, port, family, socket.SOCK_STREAM):
File "/usr/lib/python3.8/socket.py", line 918, in getaddrinfo
    for res in _socket.getaddrinfo(host, port, family, type, proto, flags):
During handling of the above exception, another exception occurred.

File "/usr/local/lib/python3.8/dist-packages/urllib3/connectionpool.py", line 703, in urlopen
    httplib_response = self._make_request(
File "/usr/local/lib/python3.8/dist-packages/urllib3/connectionpool.py", line 398, in _make_request
    conn.request(method, url, **httplib_request_kw)
File "/usr/local/lib/python3.8/dist-packages/urllib3/connection.py", line 239, in request
    super(HTTPConnection, self).request(method, url, body=body, headers=headers)
File "/usr/lib/python3.8/http/client.py", line 1256, in request
    self._send_request(method, url, body, headers, encode_chunked)
File "/usr/lib/python3.8/http/client.py", line 1302, in _send_request
    self.endheaders(body, encode_chunked=encode_chunked)
File "/usr/lib/python3.8/http/client.py", line 1259, in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
File "/usr/lib/python3.8/http/client.py", line 1071, in _send_output
    self.send(msg)
File "/usr/lib/python3.8/http/client.py", line 957, in send
    self.connect()
File "/usr/local/lib/python3.8/dist-packages/urllib3/connection.py", line 205, in connect
    conn = self._new_conn()
File "/usr/local/lib/python3.8/dist-packages/urllib3/connection.py", line 186, in _new_conn
    raise NewConnectionError(
```

On peut y lire le flag

Flag : GCSC2022{Ut1llisseZ_t0ujoUr\$_c0ll4boRat0R}

United States

Ici, on a trois fichier, une clé privée et publique RSA, et un message chiffré. On peut déchiffrer ça étant donné que nous avont les clés : <https://8gwifi.org/rsafunctions.jsp>

Generate RSA Key Size ○ 512 bit ○ 1024 bit ○ 2048 bit ○ 4096 bit

○ Encrypt to RSA Encryption

● Decrypt RSA Message

Public Key

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQC3Coz02BrFQ42/fNEfHyls569Z
0oIFZVY+Y6ppnV5/LqUoI/OUTSYBLSP1Gi2
HTikYu/Z9Rng59gkftbaxVXk/bz5
NHieAKaXdGWrW9QldGGJ1doQ8IQiMcZeb
Q3xmhJ05Uo5SFs1Fwa7mpR55e+EinNc
gT+1BqjibAcLljX18IQIDAQAB
-----END PUBLIC KEY-----
```

Private Key

```
wJBAJl3gbz4pSHqeZj1rK/Bf4lpwho8
mXHp/i9QwNta18PqtsattklNGeiJlrYEmW5
u4RxtctG14PzVzg1rvJPY5O8CQDe1
r+rTInEYXlvr7sHHaKK+ARPXzBj9DtRnAEc
NFQPFR872p2DWDLR9TS/yzNitPUMr
kXz9EtsmPm+BEijOOokCQHgQWr+oUXQ
/mhZ9mG2jZeJxfmPSof58SxG4KwFYvmX
MPTO6kbiKZ4/CLPiNqqliR1zu0i3quKaS48
w4dWVz+4=
-----END RSA PRIVATE KEY-----
```

ClearText Message

```
UxL8XXqLetXJ0h7RTifRCiKBv7zJw7siJ7ZEkw9
0+XcXqb9cezj9Ps3LFyZSjqUVIIWS0l+i2oqgk
YTaSVH6NnPOOf1B/4ulEoZfXQ8S9oSx32/2R
39ZKjN5ApplMY63AvV4U9+yV7wC1suOp9A
2LMRpRc2lvO90+FNTLhkfB7c=
```

output

```
GCSC2022{RSA_cetait_facile_quand_meme}
```

Flag : GCSC2022{RSA_cetait_facile_quand_meme}

Argentina

Il suffisait de cliquer sur “Free hint”

Flag : GCSC2022{For_your_effort}

China

Ici, on avait deux fichiers :

1) Un script python

2) Un poème

Le script :

```
1  #
2  #Trouve lerreur dans un script et exécute-le pour avoir le flag
3  #
4  #! /usr/local/bin/python  -*- coding: UTF-8 -*-
5  poeme = open('poem-14février.txt')
6
7  flag=""
8  for vers in poeme:
9      vers=vers.strip()
10     lettre = line[0:1]
11     flag=flag+lettre
12
13 print flag|
```

Il y avait une erreur dans le script à la ligne 10

```
1  lettre = line[0:1]
```

Effectivement, `line` est inconnu, il fallait le remplacer par `vers`.

J'ai aussi convertie le fichier pour qu'il soit fonctionnel sous Python3, ce qui nous donne :

```

1  #!/usr/bin/python3
2  # -*- coding: UTF-8 -*-
3  poeme = open('poem-14février.txt')
4
5  flag=""
6  for vers in poeme:
7      vers=vers.strip()
8      lettre = vers[0:1]
9      flag=flag+lettre
10
11 print(flag)

```

On peut donc maintenant le lancer et récupérer le flag :

```

kuniko@metal-age:~$ python3 script.py
FLAGCMPHDDSQNUCCPNNSOQACJOOP
kuniko@metal-age:~$ |

```

Flag : FLAGCMPHDDSQNUCCPNNSOQACJOOP

France

Ce flag était caché dans un .js (jquery), on pouvait le récupérer en entier avec quelque grep, sed etc :

```

1 curl http://challenges.guinean-cybertaskforce.com:8000/jquery-3.6.0.1.
    js -s |grep --color -P -o 'flag\+?\=\\"(.*)\\\"' |cut -d "=" -f2 |sed
    's///g' |tr -d "\n\r" && echo -e "\r"

```

Ce qui nous donne :

```

kuniko@metal-age:~$ curl http://challenges.guinean-cybertaskforce.com:8000/jquery-3.6.0.1.js -s |grep --color -P -o 'flag\+?\=\\"(.*)\\\"' |cut -d "=" -f2 |sed 's///g' |tr -d "\n\r" && echo -e "\r"
GCSC2022{n3_c0nf13Z_pas_Au_L18R4r13S}
kuniko@metal-age:~$ |

```

Flag : GCSC2022{n3_c0nf13Z_pas_Au_L18R4r13S}

Greenland

Ce flag était très long à récupérer, il fallait faire du bruteforce (dans ce cas essayer lettre par lettre) sur un service et selon la réponse essayer d'autre combinaison, par exemple :

```
kuniko@metal-age:~$ nc -nv 159.65.242.247 9090
Connection to 159.65.242.247 9090 port [tcp/*] succeeded!
A
Unexpected data
kuniko@metal-age:~$ nc -nv 159.65.242.247 9090
Connection to 159.65.242.247 9090 port [tcp/*] succeeded!
W
Flag:W
kuniko@metal-age:~$ |
```

Et petit à petit on découvre quel est le flag. J'ai donc fait un script python qui le fait automatiquement:

```
1 import sys
2 import time
3 import string
4 import socket
5
6 host = "159.65.242.247"
7 port = 9090
8
9 count = 0
10 sol = ""
11
12 while True:
13
14     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     sock.connect((host, port))
16     sock.send(sol.encode() + string.printable[count].encode() + b"\n")
17     print("Flag : " + sol + string.printable[count], end="\r")
18     response = sock.recv(1024).decode()
19     #print(response, end="\r")
20     if "Flag" in response:
21         count = 0
22         sol += response.split(":")[1][-1]
23     else:
24         count += 1
25     sock.close()
```

```
kuniko@metal-age:~$ python3 bruteforce.py
Flag : W3_h0ld_th3se_7ruths_to_be_self-3vident,_thut_all_m3n_are_creat3d_equal,_thAt_th3y_are_endowed_by_their_CreaTor_with_c3rtain_Unalienable_Rights,_that_among_th3se_4re_Life,_Lib3rty_and_the_pUrSuit_0f_Hap
piness.GCSC{H4ckInG_e7_l1b3rty}
kuniko@metal-age:~$ |
```

Ce qui sort: W3_h0ld_th3se_7ruth\$_to_be_self-3v1dent,_th4t_all_m3n_are_cre4t3d_equ4l
,_thAt_th3y_are_end0wed_by_the1r_CreaTor_with_c3rtain_Unalienable_Right5
,_that_am0ng_th3se_4re_Life,_Lib3rty_and_the_pUrsuit_0f_Happiness_GCSC{
H4cK1nG_e7_l1b3rte}

Flag : GCSC2022{H4cK1nG_e7_l1b3rte}