CTF GEMASTIK 2022

AVERAGE AMIKOM ENJOYERS

Muhammad Ichwan
Paska Parahita
Muhammad Arrya Zhahir Zain

TABLE OF CONTENT

FORENSIC 3

Traffic Enjoyer 3

Har 6

REVERSE ENGINEERING 8

Dino 8

Rubyte 10

CodeJugling 11

WEB 13

Fast Miner 13

FORENSIC

Traffic Enjoyer

Soal :

P balap first blood File: traffic.pcap

Solving Scenario :

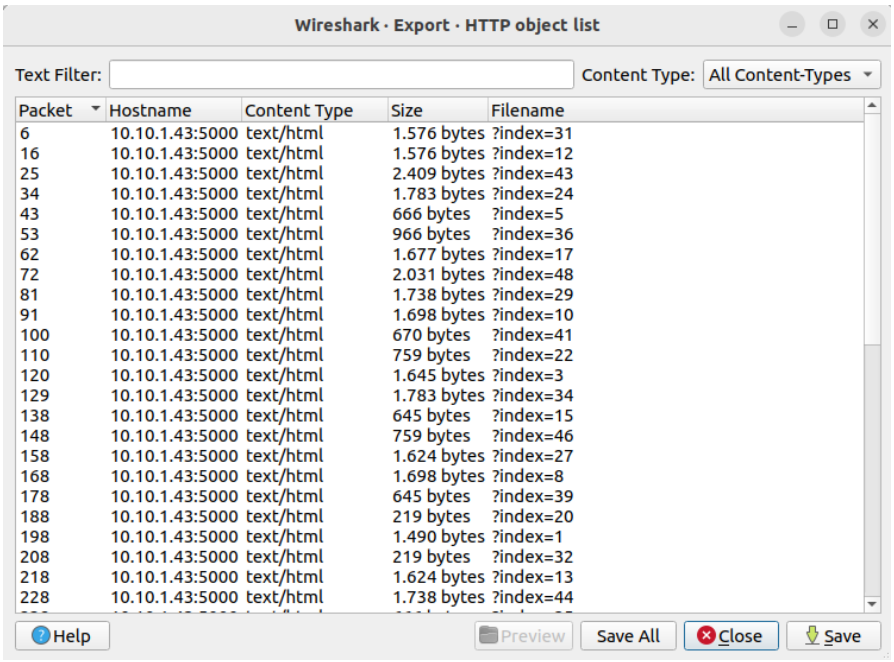
Diberikan file pcap, ketika dibuka terdapat protocol TCP dan IPA. Pada tiap packet tcp terdapat request dengan method get dan parameter *index*= , index disini increment dimulai dari 0 hingga 49, sedangkan responsenya yaitu berupa base64 yang jika didecode menghasilkan png.

```
GET /?index=0 HTTP/1.1
Host: 10.10.1.43:5000
User-Agent: Gemasteg2022
Accept-Encoding: gzip, deflate, br
Accept: */*
Connection: keep-alive

HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.8.10
Date: Sun, 30 Oct 2022 04:54:12 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1661
Connection: close
```


```
iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAIAAAD/gAIDAAAEkElEQVR4nO3cP0gy
fxwHcI28KKk4o8gi0ELqqKGI0MrShigaGoKCxugPQUtLS0MOra3BmTjWVBA2FLgE
EiaVqAS2JIkVFZkmdfjfZ3h+NPzuTL/n/el5ns9r/OB9/HzfeeLZ95RIAAAAAAAA
AAAAAAAAAAAAwJ9DKvDzyWQyg8Gg1+t7e3tVKpVSqZTL5TKZLB6PRyKRYDDo9/td
Ltfx8fHDw4PAs/0g7e3t29vb0Wg0V4RsNutwOGZmZsrKysQeXFi1tbUkSabT6WJi
+h+3293f3y/2CoTS19cXDAZZxPQlnU6vr6+LvQ7+jY2NfX5+lpLU152dHalU6H
4Wi1Wq6S+m1ra0vcFfH1t8Jx3Ov1trS05HtAPB53Op1erzccDlMUVVdXp1Qqh4eH
29ravmk7PT29v7/Pw7yislgs+V4ggUBgcXGxoqKC8cCOjo69vb1MJsN4bDgcxnFc
4LXwiyCIfKvd3d2trKws2GF8fDwWi/3Mk5FjJEkyrtNqtRbfRKvVJpNJepNYLFZT
U8Pf8ILCMIzxk6fP58t36uWztrbGGPr8/DxPwwttdHSUcYVTU1OorTAMe3x8pLey
2Wx8TC4Ck8lEX14oFGJ34cLY7e3tjfOxi8H9lVd3dze9eHp6ms1mWXSz2+30Io7j
ra2tLLqViPuwVCoVvXh+fs6u29XVVTKZpNfVajW7hqXgPqyGhgZ68fn5mV23RCLB
eGxTUxO7hqXgPqyqqip6sZR3mUgkQi/K5XLWDVkr57yjTCajFxlPpSItLS3RP7X7
/X7WDVnjPizOuVwusUf4zz/2PWRpICwEEBYCCAsBhIUAwkIAYSGAsBBAWAggLAQQ
FgKRrw2NRmNXVxeLAx0Oh9fr5XweoX18fNC/CNbr9YwPzvd/oIJWV1eFXZZEAqch
EggLAYSFAMJCAGEhEDms5eVlaSFnZ2fiDvkFXlkIICwEwEEBYCCAsBhIUAwkLAfVip
VIpeLGYfaT4YhtGLmUyGdUPWuA+Loih6sZQtxgqFoshn4Rv3Yb28vNCLzc3Nrbvw
19fTi6Js/uM+rLu7O3pxcHCQXTeCIBj3JgeDQXYNS8F9WB6Ph140GAyYM
0IvpdPrm5oZFtx8n327lubk51FZSqfT6+pre6vLyko/JRYBh2Pv7O32Ft7e3qJv9
Z2dnGXM3mUw8DS8Cs9nMuMiDg4PiN3hrNBrG0LPZrEaj4XV+QXV2dua7d8dms1VX
VxfsoNPpnp6eGDscHh7yvwJhWa1WxqmUw7Xmcrn7+/uVlRXGfboSiUSlUpEkyXjXTi6X
S6VSBEEIvJYvfN1vqFAofD7fNx+vEom68URbYqNfrv89iY2Nj
c3OTh3nFNjAwQFFUvtcXC0dHR3/zzfgTExNc5WW320u5wPwz6HS6UChYlJms51x
e/1fCMdxi8XC7ncdAoHA50Sk2CsQHEEQJEkyfnRi5Ha7FxYW2F0k8UToxX0rAMMxo
NA4NDfX09KjV6t+/RVNeXh6Px6PRaCgU8vv9FxcXJycngUBA4NkAAAAAAAAAAAAA
AAD/ql8DvrD4r8027AAAAABJRU5ErkJggg==
```


Disini kita bisa melakukan export object http pada wireshark untuk mendapatkan seluruh response base64nya disertai dengan nama file urutan index.




```
Shell-$ ls
'%3findex=0'   '%3findex=18'  '%3findex=27'  '%3findex=36'  '%3findex=45'
'%3findex=1'   '%3findex=19'  '%3findex=28'  '%3findex=37'  '%3findex=46'
'%3findex=10'  '%3findex=2'   '%3findex=29'  '%3findex=38'  '%3findex=47'
'%3findex=11'  '%3findex=20'  '%3findex=3'   '%3findex=39'  '%3findex=48'
'%3findex=12'  '%3findex=21'  '%3findex=30'  '%3findex=4'   '%3findex=49'
'%3findex=13'  '%3findex=22'  '%3findex=31'  '%3findex=40'  '%3findex=5'
'%3findex=14'  '%3findex=23'  '%3findex=32'  '%3findex=41'  '%3findex=6'
'%3findex=15'  '%3findex=24'  '%3findex=33'  '%3findex=42'  '%3findex=7'
'%3findex=16'  '%3findex=25'  '%3findex=34'  '%3findex=43'  '%3findex=8'
'%3findex=17'  '%3findex=26'  '%3findex=35'  '%3findex=44'  '%3findex=9'
```

Setelah itu kami membuat script untuk melakukan decode kemudian menyimpannya ke dalam png dan menggabungkan semua output png tersebut menjadi satu file gambar.

```python
import os
import sys
from PIL import Image

for i in range(0, 50):
  dec = os.system(f"cat \'%3findex={i}\' | base64 -d > {i}.png")

list = []
for i in range(0,50):
  list.append(f"{i}.png")

images = [Image.open(x) for x in list]

widths, heights = zip(*(i.size for i in images))

total_width = sum(widths)
max_height = max(heights)

new_im = Image.new('RGB', (total_width, max_height))

x_offset = 0
for im in images:
  new_im.paste(im, (x_offset,0))
  x_offset += im.size[0]

new_im.save('flag.png')
```

Jalankan script tersebut kemudian flag yang telah digabungkan menjadi satu kami dapatkan

```
Gemastik2022{balapan_f1rst_bl00d_is_real_f580c1763}
```

Flag :

Gemastik2022{balapan_f1rst_bl00d_is_real_f580c1763}

Har

Soal :

Har Har Har! File: har.zip

Solving Scenario :

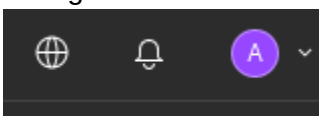
Diberikan file har, file har ini merupakan archive yang berformat file JSON, ini menyimpan data session pada banyak browser antara client dan server. Di dalam file ini berisi session figma, dan terdapat email akun figma dari probset yaitu vidnerian@mail.ru.


```
"content": {
  "size": 3367,
  "mimeType": "application/json",
  "text": "{\"error\":false,\"status\":200,\"meta\":{\"users\":[{\"id\":\"1168454184006396245\",\"name\"
  :\"vidner\",\"email\":\"vidnerian@mail.ru\",\"handle\":\"vidner\",\"img_url\":\"https://www.gravatar.c
  om/avatar/101aa999806369263988ea8556197a65?size=240&default=https%3A%2F%2Fs3-alpha.figma.com%2Fstatic%
  2Fuser_v_v2.png\",\"created_at\":\"2022-10-30T07:39:46.291Z\",\"email_validated_at\":\"2022-10-30T07:4
  4:14.112Z\",\"unsubscribed_at\":null,\"utc_offset\":null,\"profile\":{\"job_title\":\"design\"},\"phon
```

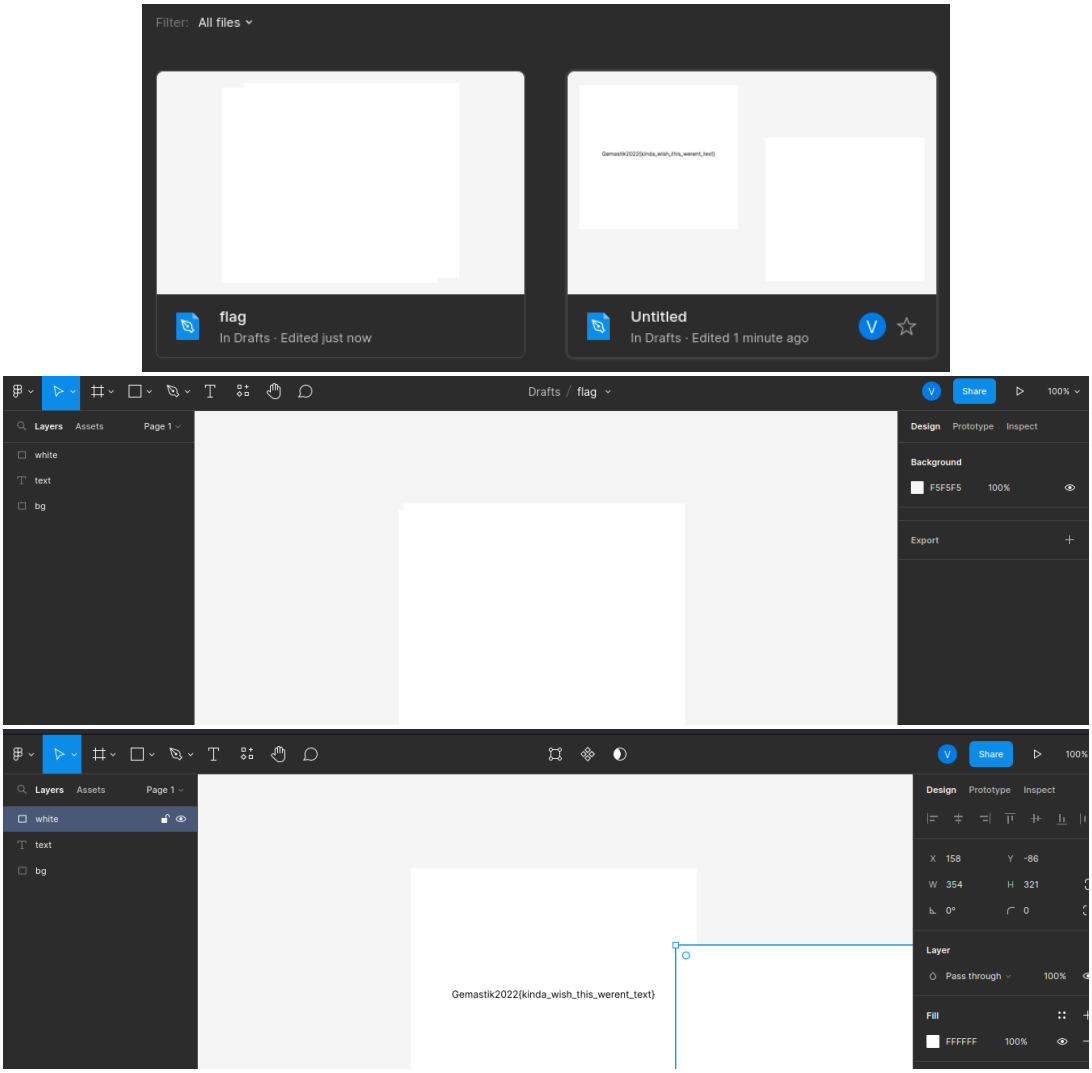

Session yang didapat yaitu milik vidner dan dengan session ini kami dapat mendapatkan akses ke dalam akun tersebut dengan cara mengubah session kami menjadi miliknya.


```
{
  "name": "cookie",
  "value": "ajs_anonymous_id=%22c72a5abf-6aa8-4c8c-8b8a-669976da42bb%22;
  _gcl_au=1.1.1443849903.1667116497; _fbp=fb.1.1667116497136.1804533433; _tt_enable_cookie=1;
  _ttp=797bd64b-84c5-4471-8c6b-773fa732cd34; cb_user_id=null; cb_group_id=null;
  cb_anonymous_id=%222b3fedda-887a-4fa1-aa29-67177b184c8e%22;
  __Host-figma.authn=%7B%221168454184006396245%22%3A%22figtkn.authn.uI5wsHnwRs0KrJjh15JdSc%22%7D;
  __Host-figma.authn-state=1;
  __Host-figma.embed=%7B%221168454184006396245%22%3A%22figtkn.embed.HPqpwVZEu8dc5v90afckEt%22%7D;
  figma.mst=1; figma.session=BAh7CEkiD3Nlc3Npb25faWQGOgZFVG86HVJhY2s6OlNlc3Npb246OlNlc3Npb25JZAY6D0Bwd
  WJsaWNfaWRJIkVjYjZjM2U0OGIwMGVlNjkwYzJhNDU3ZTdlYmNhNWRiYzRjMDY0N2YxNjQ1NzY3N2JhOWE5MzViYTY3YWFkYTA0B
  jsARkkiCmZsYXNoBjsARnsASSINdXNlcm5hbWUGOwBGSSIWdmlkbmVyaWFuQG1haWwuc
  nUGOwBU--920c4e0c9891d85c813e2d84602bb79e013ebce3; local_experiments=%22e30=%22; recent_user_data=%2
  2eyJjb21tdW5pdHlVc2VySWQiOm51bGwsImNvbW11bml0eVByb2ZpbGVJZCI6IjExNjg0NTQxODQyNjU2MzUzNjAiLCJmaWxlQnJ
  vd3NlclVzZXJJZCI6IjExNjg0NTQxODQwMDYzOTYyNDUiLCJ1c2VySWRUb09yZ0lkIjp7IjExNjg0NTQxODQwMDYzOTYyNDUiOm5
  1bGx9fQ==%22;
  AWSALBTG=rcD26NXkxhipfOt8yxZTiADAJf2jmPQ8uyJVgGiY3mAYLlNLx1iOGUG3TD3n71uDZfNkdLCaDyfYqW15tShlQliF/
  w4owouj+YeOh6Hgvj5ssrwyHwTm2vfSCSURMIxuDBxHFOGLdllzxX32ZIdjcB/phSuEm93XSyxTRhvVJ4zM; AWSALBTGCORS=rc
  D26NXkxhipfOt8yxZTiADAJf2jmPQ8uyJVgGiY3mAYLlNLx1iOGUG3TD3n71uDZfNkdLCaDyfYqW15tShlQliF/
  w4owouj+YeOh6Hgvj5ssrwyHwTm2vfSCSURMIxuDBxHFOGLdllzxX32ZIdjcB/phSuEm93XSyxTRhvVJ4zM; AWSALB=KTdlCkut
  FESozx0epZfoA6+ogbjxRYuV5jugCW8eBLUJh4xDvzr8xcyHrFpkgqBST284eXCHcYGfQ2Mr2Wl9ts80cOi8sHgVHYF1c+Iohacc
  W0oL8nj1evwZ5mF7; AWSALBCORS=KTdlCkutFESozx0epZfoA6+ogbjxRYuV5jugCW8eBLUJh4xDvzr8xcyHrFpkgqBST284eXC
  HcYGfQ2Mr2Wl9ts80cOi8sHgVHYF1c+IohaccW0oL8nj1evwZ5mF7; _ga=GA1.2.1293661695.1667116627;
  _gid=GA1.2.616608719.1667116627; _gat_UA-53594911-1=1"
},
```


Sebelumnya login dahulu dengan akun figma kita.



Kemudian ubah cookienya dengan milik akun vidner menggunakan add on cookie editor.
Setelah itu reload dan berhasil masuk ke akun vidner
Pada recents terdapat project dengan nama flag, buka kemudian terlihat beberapa layer lalu drag layer white dan didapatkan flagnya.



Flag :

Gemastik2022{kinda_wish_this_werent_text}

REVERSE ENGINEERING

Dino

Soal :

Beat my highscore! File: highscore.txt & dino.jar

Solving Scenario :

Pada file dino.jar tersebut kami melakukan decompile menggunakan tools jadx-gui. Disini terdapat 3 fungsi yang penting yaitu private void gf(), private int ls(), dan private rcr(int i). Namun untuk mendapatkan flag nya bergantung pada fungsi private int ls() ini.

```
private int ls() {
    gf();
    try {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("highscore.txt"));
        String readLine = bufferedReader.readLine();
        bufferedReader.close();
        String[] split = readLine.split(" ");
        int parseInt = Integer.parseInt(split[0]);
        this.csss = split[1];
        int rcr = rcr(parseInt);
        if (!Integer.toHexString(rcr).equals(this.csss)) {
            throw new Error("Invalid checksum");
        }
        this.ssss = rcr(rcr(rcr) ^ parseInt);
        return parseInt;
    } catch (Exception e) {
        System.out.println("Error loading highscore");
        System.exit(0);
        return 0;
    }
}
```

Pada fungsi private int ls() tersebut, file highscore yang berisi score serta hex di masukkan, kemudian file tersebut di split menjadi 2 bagian. Int parseInt berisikan score sedangkan var csss berisikan hex. Pada var rcr tersebut akan memanggil fungsi rcr dengan inputan score nya, kemudian akan dilakukan validasi dimana apabila hex dari int rcr tidak sama dengan hex dari highscore maka akan invalid. Jadi agar highscore nya kecil, kami disini membuat score baru dan juga hex nya sebagai validasi agar dapat berjalan. Untuk membuat score baru tersebut kami melakukan rebuild fungsi rcr menggunakan python.

```python
def rcr():
    var = 1
    i2 = 0
    i3 = -1
    for i4 in range(4):
        i3^= var >> (i4 * 8)
        for i5 in range(8):
            if ((i3 & 1)==1):
                i2 =(i3 >> 1) ^ -306674912
            else:
                i2 = i3 >> 1
            i3 = i2
    return i3

print(rcr())
```

Highscore kami rubah dari 2147482310 menjadi 1, dapat dilihat pada script diatas pada variable var. Kemudian untuk validasi hex nya tinggal jalankan script tersebut, kemudian convert output nya menjadi hex lalu tinggal rubah file highscore.txt nya menjadi “1 a06002d”

```
Shell-$ python3 solver.py
168165421
[23:57:50]-banua@banua:~/Desktop/GEMASTIK/REV/DINO
Shell-$ python3
Python 3.10.6 (main, Aug 10 2022, 11:40:04) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> hex(168165421)
'0xa06002d'
```

Setelah itu tinggal running jar file nya dengan java -jar dino.jar kemudian jalankan hingga score melebihi 1 lalu tabrakan maka flag di dapatkan.

Flag :

Gemastik2022{why_would_you_ever_beat_me}

Rubyte

Soal :

Hope you find the hidden gem!

File: byte.txt & output.txt

Solving Scenario :

Disini kami melakukan anlisa file byte.txt tersebut, untuk hasil analisanya kurang lebih begini:
Melakukan file read flag kemudian unpack(“H*”) index 0 dan sebagai hexadecimal, kemudian ini dilakukan secara dua kali dan di xor satu sama lain, setelah itu dilakukan bitwise. Setelah melakukan analisa code nya, kami berasumsi ini merupakan Gray Code, setelah melakukan browsing didapatkan artikel yang mirip yaitu pada (https://stackoverflow.com/questions/26481573/reversing-xor-and-bitwise-operation-in-python)
Kemudian kami tinggal ambil fungsi gray2binary dan menjalankannya dengan inputan data dari output.txt, setelah itu tinggal long_to_bytes output nya dari big int tersebut dan didapatkan flag nya.

```
from Crypto.Util.number import *
data = 215399763437993922857257938507183571899033473988099831289577921701237839559370177126393638370659139

def gray2binary(x):
    shiftamount = 1;
    while x >> shiftamount:
        x ^= x >> shiftamount
        shiftamount <<= 1
    return x

print(long_to_bytes(gray2binary(data)))
```

```
[07:09:58]-banua@banua:~/Desktop/GEMASTIK/REV/Rubybyte
Shell-$ python3 solver.py
b'Gemastik2022{i_still_remember_30_october}'
```

Flag :

Gemastik2022{i_still_remember_30_october}

CodeJugling

Soal :

Find the flag! File: reversing-itu-mudah

Solving Scenario :

Diberikan file reversing-itu-mudah, untuk melakukan analisa disini kami menggunakan ida64. Kami melakukan decompile fungsi main yang isinya sebagai berikut.

```
31    sub_401A60(a2[1], 23LL);
32    sub_401AA0(a2[1], 24LL);
33    sub_401AE0(a2[1], 25LL);
34    sub_401B20(a2[1], 26LL);
35    sub_401B60(a2[1], 27LL);
36    sub_401BA0(a2[1], 28LL);
37    sub_401BE0(a2[1], 29LL);
38    sub_401C20(a2[1], 30LL);
39    sub_401C60(a2[1], 31LL);
40    sub_401CA0(a2[1], 32LL);
41    sub_401CE0(a2[1], 33LL);
42    sub_401D20(a2[1], 34LL);
43    v4 = 0;
44    for ( i = 0; i < 35; ++i )
45      v4 |= dword_404050[i];
46    if ( strlen(a2[1]) != 35 )
47      v4 = 1;
48    if ( v4 )
49      printf("Sorry, wrong flag\n");
50    else
51      printf("Congratulations, the flag is: %s\n", a2[1]);
52  }
53  else
54  {
55    printf("Usage: %s flag\n", *a2);
56  }
57  return 0LL;
```

Pada fungsi main, terdapat fungsi yang lumayan banyak, dimana keseluruhan dari fungsi tersebut adalah perbandingan. Ada 2 perbandingan yaitu secara langsung dan juga hasil dari XOR, karena kita tidak tau inputan berapa yang menghasilkan nilai XOR sesuai dengan fungsi tersebut, maka kami melakukan XOR bilangan hasil XOR dengan nilai XOR nya. Mungkin untuk memperjelas disini kami sertakan screenshoot kedua fungsi tersebut yang mewakili.

```
1 __int64 __fastcall sub_4014A0(__int64 a1, int a2)
2 {
3   __int64 result; // rax
4
5   result = a2;
6   dword_404050[a2] = (*(char *)(a1 + a2) ^ 0xEC) != 171;
7   return result;
8 }
```

```
__int64 __fastcall sub_4014E0(__int64 a1, int a2)
{
  __int64 result; // rax

  result = a2;
  dword_404050[a2] = *(char *)(a1 + a2) != 101;
  return result;
}
```

Langsung saja kami bikin solver nya dengan mengambil nilai dari setiap fungsi tersebut kemudian build menggunakan python, jalankan dan flag nya didapatkan

```
flag = ""
rev = [0xEC ^ 171, 101, 109, 97, 0x6C ^ 31, 0xF8 ^ 140, 0x58 ^ 49, 0x6F ^ 4, 0x37 ^ 5,
       0xCD ^ 253, 0x3E ^ 12, 0xCC ^ 254, 0x70 ^ 11, 115, 0x24 ^ 80, 0x60 ^ 84, 0x10 ^ 37,
       105, 0xC3 ^ 150, 110, 95, 77, 0x86 ^ 202, 0x80 ^ 199, 0xD8 ^ 135, 0x82 ^ 233,
       0x27 ^ 23, 0x9B ^ 172, 0x93 ^ 242, 0x7A ^ 37, 98, 52, 114, 0xD1 ^ 132, 0xD ^ 112]

for i in rev:
    flag += chr(i)
print(flag)

#FLAG = Gemastik2022{st45iUn_MLG_k07a_b4rU}
```

Flag :

Gemastik2022{st45iUn_MLG_k07a_b4rU}

WEB

Fast Miner

Soal :

Restarts every 1 hour

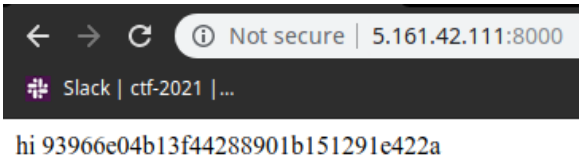
- Server 1: http://108.137.176.116:8000/
- Server 2: http://5.161.42.111:8000/

Solving Scenario :
Setelah membaca source code dari soal kami menyadari proses nya adalah

1. mengambil ID dari index
2. membuat notes
3. mendapatkan notes dengan method GET

Untuk mendapatkan flag, harus mengirim inputan sha256(2 bytes id + title), dan return nya harus 00000000 sehingga bisa mendapatkan flag yang sudah di set. Permasalahan ini berkaitan dengan POW (proof of work) bitcoin.
Kami melakukan download semua file block bitcoin di https://gz.blockchair.com/bitcoin/blocks/ dengan wget
wget -r -np -R "index.html" -e robots=off https://gz.blockchair.com/bitcoin/blocks/*

1. Kunjungi index web



Kemudian script ini kami dapatkan dari https://github.com/HITB-CyberWeek/hitbsecconf-ctf-2022/blob/main/sploits/kv/exploit.py, dengan melakukan sedikit perubahan
hash prefix diambil berdasarkan 4 byte pertama dari index

```
import csv
import datetime
import gzip
import hashlib
import os
import pytz
import urllib.parse
```

```
from binascii import unhexlify, hexlify

# BLOCKS_DIR = "blocks/"
BLOCKS_DIR = "gz.blockchair.com/bitcoin/blocks/"

def swap_endianness(data: str) -> str:
    assert len(data) % 2 == 0, f"data should have even length, but given: {data}"
    return "".join(data[idx:idx+2] for idx in range(len(data) - 2, -1, -2))


def calculate_block_hash(previous_block_hash: str, block: dict) -> str:
    version = int(block["version"])
    merkle_root = block["merkle_root"]
    nonce = int(block["nonce"])
    bits = int(block["bits"])
    time = datetime.datetime.strptime(block["time"], "%Y-%m-%d %H:%M:%S")
    timestamp = int(time.replace(tzinfo=pytz.UTC).timestamp())

    header_parts = [
    "%08x" % version,
    previous_block_hash,
    merkle_root,
    "%08x" % timestamp,
    "%08x" % bits,
    "%08x" % nonce
    ]
    header_hex = "".join(map(swap_endianness, header_parts))
    header_bin = unhexlify(header_hex)
    calculated_hash_a = hashlib.sha256(header_bin).digest()
    calculated_hash_b = hashlib.sha256(calculated_hash_a).digest()

    return hexlify(calculated_hash_a).decode(), hexlify(calculated_hash_b).decode(),
hexlify(calculated_hash_b[::-1]).decode()

def find_block(prefix):
    previous_block_hash = None

    for filename in os.listdir(BLOCKS_DIR):
    filepath = os.path.join(BLOCKS_DIR, filename)
    # print(filepath)
    with gzip.open(filepath, 'rt') as file:
    tsv_file = csv.DictReader(file, delimiter="\t")
    for block in tsv_file:
```

```
        block_hash = block["hash"]

        try:
        if previous_block_hash is None:
        continue

        if not block_hash.startswith('0' * 16):
        continue

        calculated_block_hash_a, calculated_block_hash_b,
calculated_block_hash_b_reversed = calculate_block_hash(previous_block_hash, block)
        # print(calculated_block_hash_a)
        if calculated_block_hash_a.startswith(prefix):
        if calculated_block_hash_b_reversed != block_hash:
                continue
        # assert calculated_block_hash_b == block_hash
        return filepath, calculated_block_hash_a, calculated_block_hash_b, block_hash

        finally:
        previous_block_hash = block_hash

hash_prefix = "9396" # 4 bytes id from index
block = find_block(hash_prefix)
filename_hex = block[1][4:]
filename = unhexlify(filename_hex)
filename_url = urllib.parse.quote(filename)
print("PREFIX:{} FILENAME_HEX:{} FILENAME_URL:{} BLOCK:{}".format(hash_prefix,
filename_hex, filename_url, block))
```

Jalankan scriptnya lalu didapatkan hasil kalkulasi

```
⇒  python3 exp.py
PREFIX:9396 FILENAME_HEX:529f9fc08f2ae786299a280bf32db47e37d58fb90529d6e80d5f
fb8f11e8 FILENAME_URL:R%9F%9F%C0%8F%2A%E7%86%29%9A%28%0B%F3-%B4%7E7%D5%8F%B9%
05%29%D6%E8%0D_%FB%8F%11%E8 BLOCK:('gz.blockchair.com/bitcoin/blocks/blockcha
ir_bitcoin_blocks_20171221.tsv.gz', '9396529f9fc08f2ae786299a280bf32db47e37d5
8fb90529d6e80d5ffb8f11e8', 'ab3e20bcf171f4eb122d6c590d311683b4cd7752b95146000
000000000000000', '00000000000000000004651b95277cdb48316310d596c2d12ebf471f1bc
203eab')
```

2. Membuat notes dengan title filename_url yang didapat dari kalkulasi sebelumnya, dan sessionId diambil dari index http://5.161.42.111:8000/ namun hasilnya Unauthorized tetapi saat kami coba akses notesnya dengan titlenya filename_url berhasil didapat flag

```
⇒  curl -X POST 'http://5.161.42.111:8000/notes' \
  -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9' \
  -H 'Accept-Language: en-US,en;q=0.9' \
  -H 'Cache-Control: max-age=0' \
  -H 'Connection: keep-alive' \
  -H 'Cookie: sessionId=9a9ce7e7-0370-46c9-ac95-d5c88f052f62' \
  --data "title=R%9F%9F%C0%8F%2A%E7%86%29%9A%28%0B%F3-%B4%7E7%D5%8F%B9%05%29%
D6%E8%0D_%FB%8F%11%E8&content=testing" \
  --compressed
unauthorized%
```

3. Mengunjungi notes dengan title sesuai dengan hasil filename_url

Gemastik2022{hopefully_no_one_should_approach_this_with_bruteforce}

Flag :

Gemastik2022{hopefully_no_one_should_approach_this_with_bruteforce}