

Write Up Gemastik 2022

HengkerNgangNgong

Dino (Reverse Engineering)

Kita diberikan sebuah file .jar (sebuah clone dari game dino google chrome) dan sebuah file bernama highscore.txt, yang menyimpan skor tertinggi dari game tersebut dan tersambung langsung ke file .jar tersebut. Objective kita adalah untuk melampaui highscore tersebut. Namun default highscore yang ada terlalu tinggi untuk dilampaui. Solusi yang kami pikirkan adalah untuk mengubah highscore yang ada menjadi lebih kecil, agar bisa dilampaui dengan mudah.

Di dalam highscore.txt, terdapat dua value yang terpisahkan oleh spasi, yaitu highscore dan sebuah bilangan hex. Kami berasumsi kedua bilangan saling berkaitan, sehingga untuk mengetahui kaitannya kami melakukan decompile terhadap file .jar.

Dengan men-decompile file .jar yang diberikan, kita menemukan terdapat 4 class dengan DinoGame.class yang menjadi main class. Setelah menelusuri DinoGame.class, terdapat sebuah fungsi bernama rcr() yang menerima sebuah argumen integer dan mengembalikan sebuah integer lain setelah berbagai operasi bitwise, yang mana jika integer tersebut diubah menjadi hex, nilainya akan sama dengan bilangan hex yang terdapat di highscore.txt.

```
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;  
}
```

Kemudian program memeriksa kesamaan antara bilangan hex yang terdapat di dalam highscore.txt dan bilangan hex yang dihasilkan dari program. Jika sama, maka program dapat berjalan. Sehingga untuk mengganti highscore di highscore.txt,

kita juga perlu mengetahui hex yang dihasilkan dari fungsi rcr() untuk highscore yang kita set.

```
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;
}
```

Solusi cukup mudah, kita hanya perlu menulis ulang fungsi rcr() (persis) dalam program Java lain dan memasukkan angka yang kita inginkan untuk menjadi highscore. Setelah mendapatkan bilangan hex yang dicek kesamaannya, dapat kita masukkan ke file highscore.txt. Program akan berjalan dengan baik dan highscore dapat kita manipulasi menjadi lebih kecil, sehingga tidak perlu terlalu lama mengumpulkan skor dan kita bisa mendapatkan flag.

Flag: Gemastik2022{why_would_you_ever_beat_me}

Har (Forensics)

Kita mendapatkan file zip yang didalamnya berisi sebuah file berekstensi .har. Kita bisa membaca file tersebut menggunakan Google HAR Analyzer.

Di HAR Analyzer, kita bisa melihat bahwa kita sedang berinteraksi dengan website Figma. Setelah kita analisa, ada satu packet yang memiliki atribut canvas_url.

Request Response **Response Content** Cookies Timing

Compressed:

No.

[MIME type:](#)

application/json

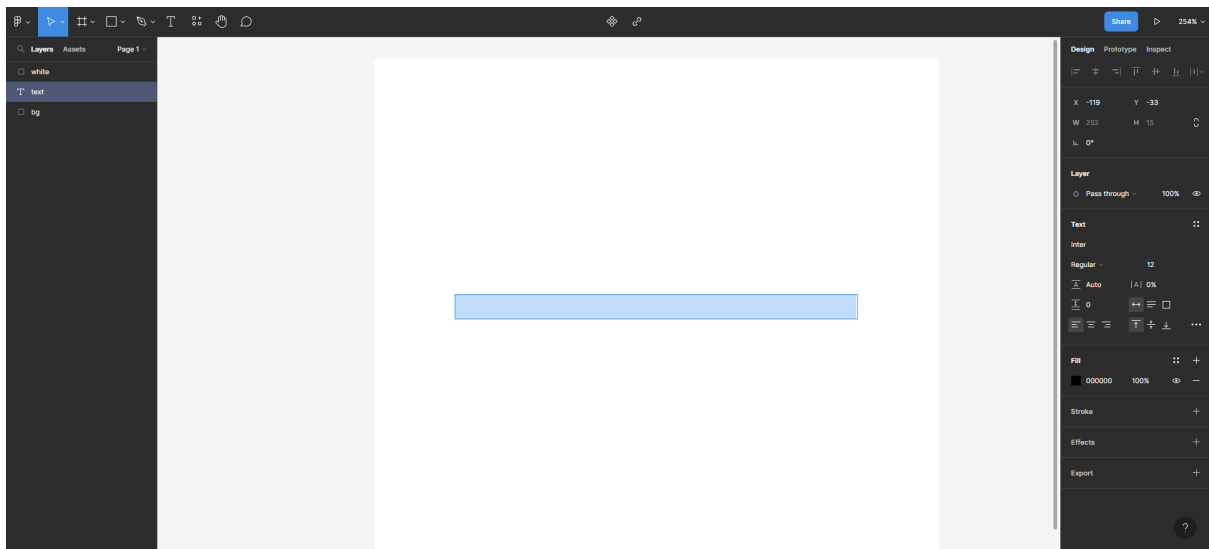
Content:

Download the received [content](#) .

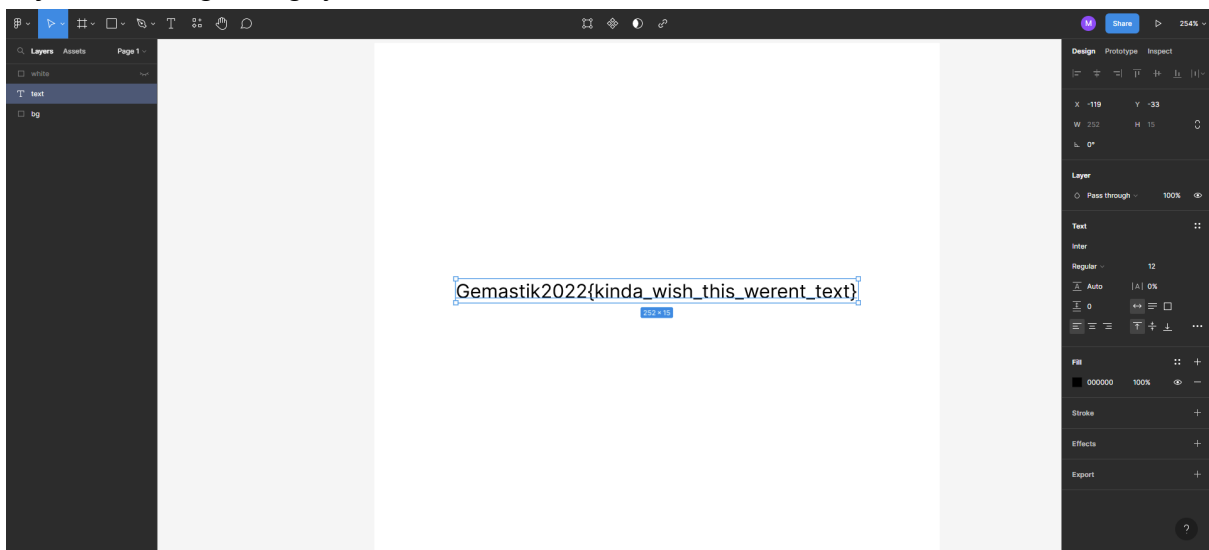
```
{
  "error": false,
  "status": 200,
  "meta": {
    "is_default": null,
    "script": "cef6a458c5668f8c7570d55230ad629630428847",
    "name": "Untitled",
    "file_key": "N2D2B1mcWjiqgKZciiPSJ5",
    "source_file_key": null,
    "file_repo": null,
    "needs_upgrade": false,
    "canvas_url": "https://s3-alpha-sig.figma.com/checkpoints/pUE/y5I/cFaNjWYebNG1JmUi/3qnr9v3zrywpThYifXxA4.fig?Expires=1667779200&Signature=U",
    "pass_state": "{\\\"file_key\\\":\\\"N2D2B1mcWjiqgKZciiPSJ5\\\",\\\"session_id\\\":281734461}",
    "realtime_token": "/file-N2D2B1mcWjiqgKZciiPSJ5:1667116666:0:084345babd34efa8f88135d868c7b9eb9d4b47bb",
    "requested_at": 1667116666,
    "org_drafts_owners": [],
    "prefix": "https://www.figma.com/file/N2D2B1mcWjiqgKZciiPSJ5/image",
    "shared_fonts": [],
    "is_community_duplicate": false,
    "feature_flags": {
      "disable_registerframecallback": true,
      "survey_pro_cart_abandon": true,
      "widgets_paste_as_widget": true,
      "community_merged_install_logic": true,
      "email_org_first_annual_renewal": true,
      "plugins_nodechange": true,

```

Jika kita buka canvas_url tersebut, kita akan mendownload sebuah file berekstensi .fig. Dimana file tersebut adalah tipe file yang digunakan oleh Figma. Jika kita import file tersebut ke Figma, kita mendapatkan sebuah project seperti gambar dibawah.



Terlihat ada object text disitu yang kemungkinan adalah flagnya. Maka, kita hide object white agar flagnya terlihat.

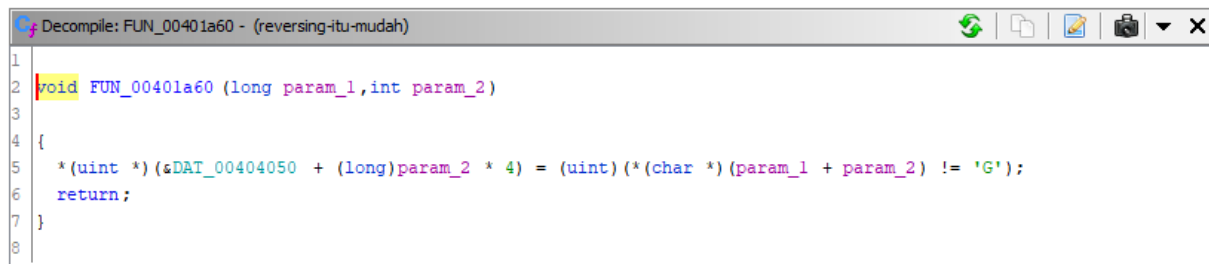


Flag: Gemastik2022{kinda_wish_this_werent_text}

Code Juggling (Reverse Engineering)

Kita diberikan sebuah file binary bernama reversing-itu-mudah. Dengan membukanya di ghidra, kita menemukan ada banyak sekali fungsi. Fungsi fungsi tersebut dipanggil secara berurutan.

Ketika kita buka function-function tersebut satu satu, kita melihat ada 1 character di setiap function yang terlihat seperti format flag.



```
Decompile: FUN_00401a60 - (reversing-itu-mudah)
1
2 void FUN_00401a60 (long param_1,int param_2)
3
4 {
5     *(uint *)(&DAT_00404050 + (long)param_2 * 4) = (uint)(*(char *) (param_1 + param_2) != 'G');
6     return;
7 }
8
```

Ketika semua character dari function-function tersebut disatukan, kita mendapatkan flagnya.

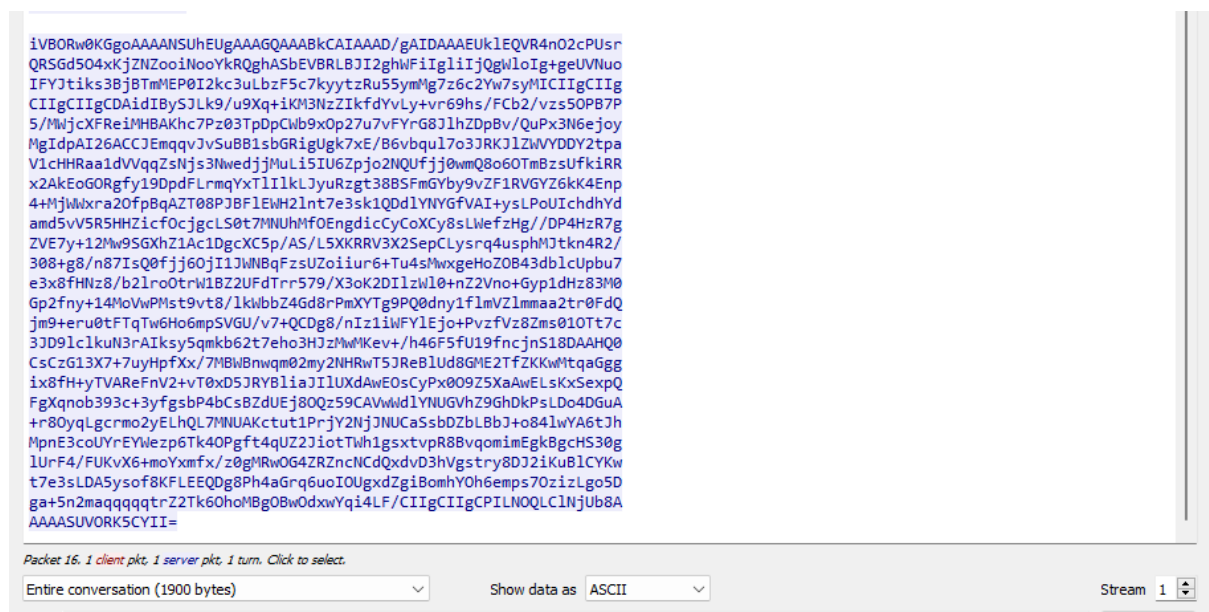
Flag: Gemastik2022{st45iUn_MLG_k07a_b4rU}

Traffic Enjoyer (Forensics)

Kita diberikan sebuah file .pcap. Dari file .pcap tersebut terdapat dua protocol yaitu TCP dan HTTP. Kita dapat melihat bahwa protocol HTTP mengirimkan teks dari html. Sehingga kita hanya perlu fokus pada protocol HTTP karena flag kemungkinan besar berada disitu.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.126.129	10.10.1.43	TCP	74	33198 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
2	0.000870	10.10.1.43	192.168.126.129	TCP	60	5000 → 33198 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
3	0.000928	192.168.126.129	10.10.1.43	TCP	54	33198 → 5000 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.001041	192.168.126.129	10.10.1.43	HTTP	203	GET /?index=31 HTTP/1.1
5	0.001340	10.10.1.43	192.168.126.129	TCP	60	5000 → 33198 [ACK] Seq=1 Ack=150 Win=64240 Len=0
6	0.002870	10.10.1.43	192.168.126.129	HTTP	1805	HTTP/1.1 200 OK (text/html)
7	0.002871	10.10.1.43	192.168.126.129	TCP	60	5000 → 33198 [FIN, PSH, ACK] Seq=1752 Ack=150 Win=6
8	0.002927	192.168.126.129	10.10.1.43	TCP	54	33198 → 5000 [ACK] Seq=150 Ack=1752 Win=62780 Len=0
9	0.003606	192.168.126.129	10.10.1.43	TCP	54	33198 → 5000 [FIN, ACK] Seq=150 Ack=1753 Win=62780
10	0.004004	10.10.1.43	192.168.126.129	TCP	60	5000 → 33198 [ACK] Seq=1753 Ack=151 Win=64239 Len=0

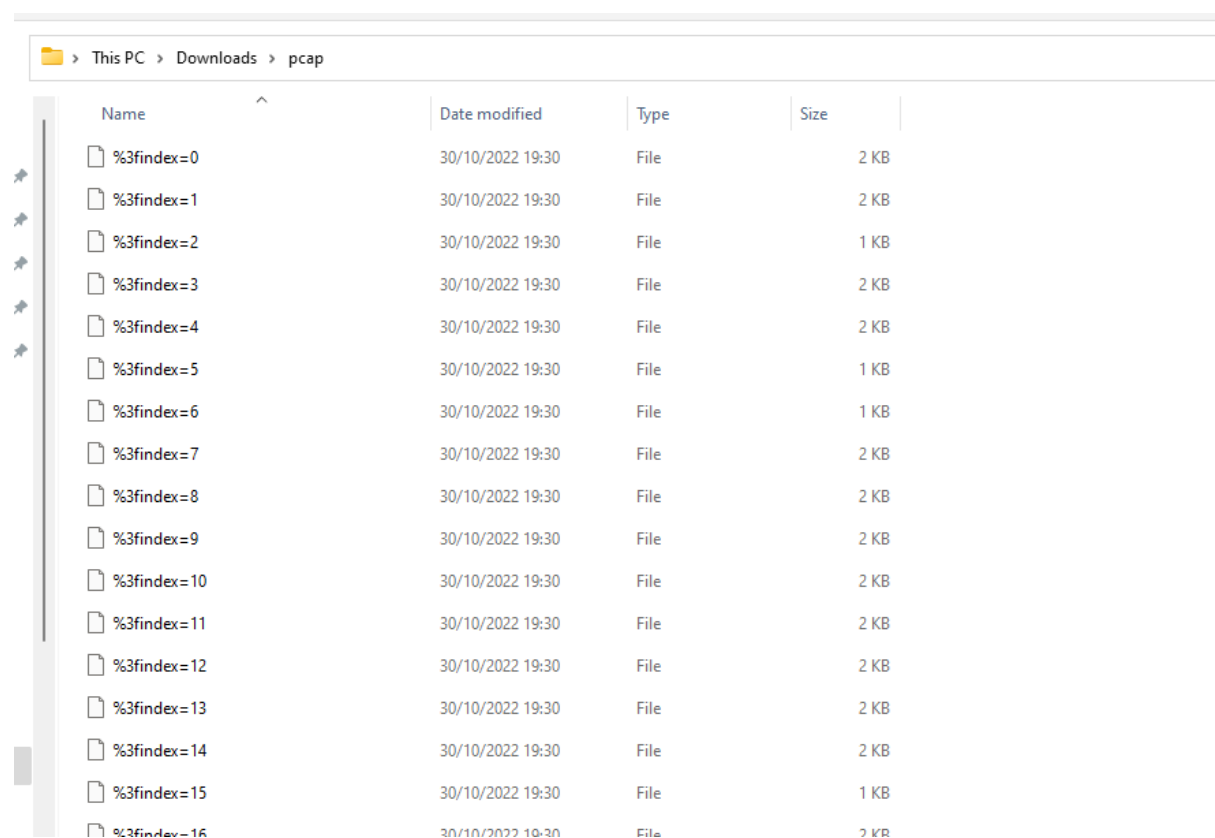
Ketika kita melihat HTTP stream melalui follow TCP stream, kita bisa melihat bahwa tiap-tiap stream mengirimkan sebuah string terenkripsi, yang mana enkripsi tersebut kami asumsikan adalah base64.



Kami mencoba men-decode string tersebut menggunakan base64, dan ternyata benar! Ditemukan bahwa string tersebut mengandung bytes yang merupakan struktur dari sebuah file .png. Jika bytes ini kami tuliskan ke file .png, tentunya akan ada sebuah gambar yang dihasilkan.



Oleh karena itu, kami mengexport semua isi stream HTTP yang ada di Wireshark (File > Export objects > HTTP > Save all) dan mengurutkannya sesuai indexnya.



#DiamMenggabutBergerakNgangNgong

Menggunakan Python, kami melakukan automasi untuk mendecode setiap isi dari stream tersebut dan menuliskannya sebagai bytes kedalam file png untuk menghasilkan file png.

```
import base64 as b

filename = r"C:\Users\nmuaf\Downloads\pcap\%3findex="

for i in range(50):
    with open(filename+str(i), "rb") as file:
        isi = file.read()
        newfile = open(rf"C:\Users\nmuaf\Downloads\pcap\gambar{i}.png", "wb")
        newfile.write(b.b64decode(isi))
```

Automasi sukses, dan kami mendapatkan 50 gambar yang mana masing-masing gambar adalah satu persatu huruf dari flagnya.



Flag: Gemastik2022{balapan_f1rst_bl00d_is_real_f580c176}