Mod:

Description

Just a simple modulo challenge

Server's code:

```
#!/usr/local/bin/python3
import os
secret = int(os.urandom(32).hex(),16)
print("Welcome to Mod!")
num=int(input("Provide a number: "))
print(num % secret)
guess = int(input("Guess: "))
if guess==secret:
    print(open('flag.txt').read())
else:
    print("Incorrect!")
```

How server worked:

- Create `secret` which was a very big positive random number
- `num` was put by user
- Print `num % secret`
- User guessed and put into `guess`
- If `guess == secret`, print flag

Solution:

I saw `num = int(input())` $\Longrightarrow$ Both positive and negative number available.

With the properties of module:
If I have `mod = a % b` $\Longrightarrow$ `b * c + mod = a` with `c = a / b`
Example : `7 % 2 = 1` because `2*(7/2) + 1 = 7`

Follow that, I thought that I could put `num = -1` and I could have `-1 % secret = mod` with `mod` printed by server and I got:

```
    -1 % secret = mod
=> mod + (-1)*secret = -1
```

```
=> secret = mod + 1
```

Put secret found and get the flag: `scriptCTF{-1_f0r_7h3_w1n_4a3f7db1_a74b01dd41e5}`

`Note`: I dont use other negative number because it will make `mod` much more difficult to take advantage of properties of module

Substract

Description

The image size is 500x500. You might want to remove some stuff... Note: Some may call it guessy!

Provided:

Challenge provided a file `coordinates.txt` with each line was a coordinates `(x,y)`

Solution

"The image size is 500x500" so I thought that I could create a full white image with that size and each coordinate in `txt` file was a black dot.
However, when I finished, I got full black image $\Longrightarrow$ There were several coordinates which duplicated. Therefore, I had to remove them:

```
from collections import Counter
def remove_all_duplicates(input_file, output_file):
    counter = Counter()
    with open(input_file, "r") as f:
        for line in f:
            line = line.strip()
            if not line:
                continue
            line = line.replace("(", "").replace(")", "")
            try:
                x, y = map(int, line.split(","))
                counter[(x, y)] += 1
            except ValueError:
                continue

    with open(output_file, "w") as f:
        for coord, count in counter.items():
            if count == 1:  # Chỉ lưu tọa độ xuất hiện đúng 1 lần
                f.write(f"({coord[0]},{coord[1]})\n")
    print(f"✅ Done. Save at {output_file}")
```

```
if __name__ == "__main__":
    remove_all_duplicates("coordinates.txt", "new_coordinatess.txt")
```

After that, I printed black dot into white image:

```
from PIL import Image, ImageDraw

def load_coordinates(filename, size=500):
    coords = []
    with open(filename, "r") as f:
        for line in f:
            line = line.strip()
            if not line:
                continue
            line = line.replace("(", "").replace(")", "")
            try:
                x, y = map(int, line.split(","))
                if 0 <= x < size and 0 <= y < size:
                    coords.append((x, y))
            except ValueError:
                continue
    return coords


def draw_coordinates(coords, size=500, outfile="output.png"):
    img = Image.new("RGB", (size, size), (0,0, 0))
    draw = ImageDraw.Draw(img)
    for x, y in coords:
        draw.point((x, y), fill=(255, 255, 255))

    img.save(outfile)
    print(f"✅ Done {outfile}, with {len(coords)} coordinates.")

if __name__ == "__main__":
    coords = load_coordinates("new_coordinates.txt", size=500)
    draw_coordinates(coords, size=500, outfile="final.png")
```

Flag printed on the received image

Flag: `scriptCTF{5ub7r4c7_7h3_n01s3}`

Sums

Description

Find the sum of nums[i] for i in [l, r] (if there are any issues with input/output format, plz open a ticket)

Server's code:

```
#!/usr/bin/env python3
import random
import subprocess
import sys
import time

start = time.time()

n = 123456

nums = [str(random.randint(0, 696969)) for _ in range(n)]

print(' '.join(nums), flush=True)

ranges = []
for _ in range(n):
    l = random.randint(0, n - 2)
    r = random.randint(l, n - 1)
    ranges.append(f"{l} {r}") #inclusive on [l, r] 0 indexed
    print(l, r)

big_input = ' '.join(nums) + "\n" + "\n".join(ranges) + "\n"

proc = subprocess.Popen(
    ['./solve'],
    stdin=subprocess.PIPE,
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    text=True
)

stdout, stderr = proc.communicate(input=big_input)

out_lines = stdout.splitlines()
ans = [int(x) for x in out_lines[:n]]

urnums = []
for _ in range(n):
    urnums.append(int(input()))

if ans != urnums:
```

```
    print("wawawawawawawawawa")
    sys.exit(1)

if time.time() - start > 10:
    print("tletletletletletle")
    sys.exit(1)

print(open('flag.txt', 'r').readline())
```

How server work

```
n = 123456
nums = [str(random.randint(0, 696969)) for _ in range(n)]
print(' '.join(nums), flush=True)
```

That code print 123456 number in range from 0 to 696969

```
for _ in range(n):
    l = random.randint(0, n - 2)
    r = random.randint(l, n - 1)
    print(l, r)
```

Print 123456 pair `(l,r)`

After that run `./solve`

Check user input:

```
urnums = []
for _ in range(n):
    urnums.append(int(input()))
if ans != urnums: fail
```

Server wait user put 123456 number if user input is the same with `ans` of `solve` → Correct

Check time:

```
if time.time() - start > 10: fail
```

All have to do in 10s

If pass, server print flag

Solution:

Server do not check any except `ans == urnums` so at that time I thought I had to know what `./solve` printed

So I could use this code:

```
# pip install pwntools

from pwn import *
import sys

HOST, PORT = "play.scriptsorcerers.xyz", 10105
n = 123456  # theo script gốc

def recv_until_have_n_ints(r):
    """Đọc nhiều dòng cho đến khi gom đủ n số ở phần nums đầu tiên."""
    buf = b""
    while True:
        # server thường gửi nums thành 1 dòng rất dài; nhưng để chắc ăn, gom
đến khi đủ n số
        chunk = r.recvline(timeout=60)
        if not chunk:
            break
        buf += chunk
        tokens = buf.split()
        if len(tokens) >= n:
            # giữ lại đúng n số đầu, phần dư (nếu có) trả lại cho stream xử
lý tiếp
            nums = list(map(int, tokens[:n]))
            leftover = b" ".join(tokens[n:]) + b"\n" if len(tokens) > n else
b""
            return nums, leftover
    raise RuntimeError("Không nhận đủ nums từ server")

def main():
    r = remote(HOST, PORT)

     1) Nhận nums (đôi khi server có banner/giới thiệu, cứ đọc cho đến khi
gom đủ n số)
    nums, leftover = recv_until_have_n_ints(r)

    # 2) Nhận tiếp n dòng ranges (có thể một phần đã nằm trong leftover)
    ranges = []
    # ưu tiên xử lý phần dư trước
    if leftover.strip():
```

```
        for line in leftover.strip().split(b"\n"):
            if not line.strip():
                continue
            l, rr = map(int, line.split())
            ranges.append((l, rr))

    while len(ranges) < n:
        line = r.recvline(timeout=60)
        if not line:
            break
        line = line.strip()
        if not line:
            continue
        l, rr = map(int, line.split())
        ranges.append((l, rr))

    if len(nums) != n or len(ranges) != n:
        print(f"Thiếu dữ liệu: len(nums)={len(nums)}, len(ranges)=
{len(ranges)}")
        r.close()
        sys.exit(1)

    # 3) Prefix sum
    prefix = [0]*(n+1)
    s = 0
    for i, v in enumerate(nums, 1):
        s += v
        prefix[i] = s

    # 4) Tính đáp án cho từng (l, r)
    out_lines = []
    for l, rr in ranges:
        out_lines.append(str(prefix[rr+1] - prefix[l]))
    payload = "\n".join(out_lines) + "\n"

    # 5) Gửi lại toàn bộ một lần (nhanh và chắc ăn)
    r.send(payload.encode())

    # 6) Nhận flag
    try:
        print(r.recvall(timeout=30).decode(errors="ignore"))
    finally:
        r.close()

if __name__ == "__main__":
    main()
```

Summarize that code:

1. Connect to server
2. Read `nums`
    1. Server printe 123456 numbers
    2. This code read all until enough `n` numbers
    3. If there is any extra reading (there may be some more l r pairs later) → return it to leftover.
3. Continue read `ranges`
    1. Read enough 123456 numbers
4. Calculate `prefix sum`
    1. `prefix[i]` = sum of `nums[0, ..., i-1]`
5. Answer query: `sum(nums[l..r]) = prefix[r+1] - prefix[l]`
6. Send results
7. Get flag

In `Terminal` tab of VSCode:

```
[x] Opening connection to play.scriptsorcerers.xyz on port 10105
[x] Opening connection to play.scriptsorcerers.xyz on port 10105: Trying
34.174.160.24
[+] Opening connection to play.scriptsorcerers.xyz on port 10105: Done
[x] Receiving all data
[x] Receiving all data: 0B
[x] Receiving all data: 42B
[+] Receiving all data: Done (42B)
[*] Closed connection to play.scriptsorcerers.xyz port 10105
scriptCTF{1_w4n7_m0r3_5um5_fad96246f38a}
```

Flag: `scriptCTF{1_w4n7_m0r3_5um5_fad96246f38a}`

pdf

Description

so sad cause no flag in pdf :(

Provided:

Challenge provided a `.pdf` file with: `thx for comming but no flag here :)`

Solution:

Because "so sad cause no flag in pdf :(" so at that time I thought I had to open this `pdf` file with other to get other file and `binwalk` was the best approach

```
mduc7@MinhDuc:/mnt/d/project/PythonProject/WU-ScriptCTF/pdf$ binwalk
challenge.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
----
0             0x0             PDF document, version: "1.4"
283           0x11B           Zlib compressed data, default compression
1521          0x5F1           Copyright string: "copyright/ordfeminine
172/logicalnot/.notdef/registered/macron/degree/plusminus/twosuperior/threes
uperior/acute/mu 183/periodcen"
```

From that result, I saw `pdf` file had other file in it so I wanted to open it

```
mduc7@MinhDuc:/mnt/d/project/PythonProject/WU-ScriptCTF/pdf$ binwalk -e
challenge.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
----
0             0x0             PDF document, version: "1.4"
283           0x11B           Zlib compressed data, default compression
1521          0x5F1           Copyright string: "copyright/ordfeminine
172/logicalnot/.notdef/registered/macron/degree/plusminus/twosuperior/threes
uperior/acute/mu 183/periodcen"
```

Added `-e` to extract `pdf` file with `binwalk` and it created a new folder which contained `zlib` file
→ I got `11B` file and `11B.zlib`, opened `11B` to get flag

Flag: `scriptCTF{pdf_s7r34m5_0v3r_7w17ch_5tr34ms}`

Enchant

Description

I was playing minecraft, and found this strange enchantment on the enchantment table. Can you figure out what it is? Wrap the flag in scriptCTF{}

Provided:

Challenge provided a file with string: `á’²â•Žãƒªá’·á“µâˆ·á”‘âŽ“â„¸ Ì£ â•Žá“âŽ“âšãƒª`

Solution

Challenge told that this string related to `Minecraft` and enchantment table. I found on Internet, the enchantment table use `Standard Galactic Alphabet (SGA)` which was a `hieroglyphic system`, each SGA character equivalent to Latinh character

So I had to convert that string into SGA → Translate SGA to Latinh → Get flag

To convert that string, I used a online tool: https://www.linestarve.com/tools/mojibake/ and default settings
$\implies$ Decoded string: ᒲ╎リᒷᓵ∷ᔑ⎓ℸ ̣ ╎ᓭ⎓⚍リ

Used other online tool to translate: https://www.dcode.fr/standard-galactic-alphabet
$\implies$ Translated string: `MINECRAFTISFUN`

$\implies$ Flag: `scriptCTF{MINECRAFTISFUN}`

Secure Server

Description

John Doe uses this secure server where plaintext is never shared. Our Forensics Analyst was able to capture this traffic and the source code for the server. Can you recover John Doe's secrets?

Server's code:

```
import os
from pwn import xor
print("With the Secure Server, sharing secrets is safer than ever!")
enc = byte.fromhex(input("Enter the secret, XORed by your key (in hex):
").strip())
key = os.urandom(32)
enc2 = xor(enc,key).hex()
print(f"Double encrypted secret (in hex): {enc2}")
dec = bytes.fromhex(input("XOR the above with your key again (in hex):
").strip())
secret = xor(dec,key)
print("Secret received!")
```

Additionally, challenge provided a `.pcap` file which included some data exchanged between server and user
File: `WU-ScriptCTF`

Solution

At that time, I saw server code was very simple with only `XOR` and 2 keys:

1. origin $\oplus$ `key_user` = `enc`
2. `enc` $\oplus$ `key_server` = `enc2`
3. `enc2` $\oplus$ `key_user` = `dec`
4. `dec` $\oplus$ `key_server` = `secret`

$\Longrightarrow$ `key_server` = `enc` $\oplus$ `enc2` and `secret` = `dec` $\oplus$ `key_server`

Open `pcap` and find the packet 6, 8, 10 had data of these strings, copied it ad `Hex dump` and pasted to `Cypherchef`

```
enc = 151e71ce4addf692d5bac83bb87911a20c39b71da3fa5e7ff05a2b2b0a83ba03
enc2 = e1930164280e44386b389f7e3bc02b707188ea70d9617e3ced989f15d8a10d70
dec = 87ee02c312a7f1fef8f92f75f1e60ba122df321925e8132068b0871ff303960e
```

$\Longrightarrow$ I used a simple Python script to find flag:

```
import pwn as xor


enc_hex  =
"151e71ce4addf692d5bac83bb87911a20c39b71da3fa5e7ff05a2b2b0a83ba03"

enc2_hex =
"e1930164280e44386b389f7e3bc02b707188ea70d9617e3ced989f15d8a10d70"

dec_hex  =
"87ee02c312a7f1fef8f92f75f1e60ba122df321925e8132068b0871ff303960e"

enc = bytes.fromhex(enc_hex)
enc2 = bytes.fromhex(enc2_hex)
dec = bytes.fromhex(dec_hex)

key_server = b''
for i,j in zip(enc,enc2):
    key_server += bytes([i^j])
print(f"Key server: {key_server.hex()}")
secret = b''
for i,j in zip(dec, key_server):
    secret += bytes([i^j])
print(f"Secret: {secret}")
```

Flag: `scriptCTF{x0r_1s_not_s3cur3!!!!}`

diskchal

Description

i accidentally vanished my flag, can u find it for me

Provided:

Challenge provided a `.img` file which was a disk image

Solution:

I had a deep see in this disk image with `fdisk`:

```
fdisk -l stick.img
```

and I got:

```
Disk stick.img: 24 MiB, 25165824 bytes, 49152 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: 0x00000000
```

Created mapping:

```
sudo kpartx -av stick.img

sudo mkdir -p /mnt/stick

sudo mount -o loop stick.img /mnt/stick2
```

After that, accessed to the folder `stick2` but nothing related to flag in here but at `random_thoughts.txt` I saw: `i wonder where i put the flag. did i palm it somewhere?`.
The word `palm` in magic related to hide something in hand so I thought I had been fooled with this approach

Accessed again to folder which saved `stick.img`, I used `hexdump` and saw below at the end:

```
00062e00  1f 8b 08 08 ca 78 79 68  00 03 66 6c 61 67 2e 74
|.....xyh..flag.t|
00062e10  78 74 00 2b 4e 2e ca 2c  28 71 0e 71 ab 36 8c cf  |xt.+N..,
(q.q.6..|
00062e20  31 28 33 8e cf 35 31 33  4c 8e 37 2f 32 4c ce 36
|1(3..513L.7/2L.6|
00062e30  ad e5 02 00 0b a1 b6 db  1f 00 00 00 00 00 00 00
|................|
00062e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
```

Based on that, I used `binwalk`:

```
binwalk stick.img
```

And got:

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
----
404992        0x62E00         gzip compressed data, has original file name:
"flag.txt", from Unix, last modified: 2025-07-17 22:27:22
```

This revealed that `stick.img` had a `.gz` file start at offset `404992` and origin name was `flag.txt` $\implies$ Clearly this was what I need to open

Used `dd` to extract `gzip` from disk image:

```
dd if=stick.img bs=1 skip=404992 of=flag.gz
```

`dd`: Command to copy low level data in Linux (WSL)
`if=stick.img`: Input file
`bs=1`: Block size - each time read is 1 byte
`skip=404992`: Skip 404992 byte at first of input file
`of=flag.gz`: Output file

After that, unzipped the `flag.gz` and opened `flag.txt`

Flag: `scriptCTF{1_l0v3_m461c_7r1ck5}`
`

RSA-1

Description

Yú Tóngyī send a message to 3 peoples with unique modulus. But he left it vulnerable. Figure out :)

Provided:

A `.txt` file:

```
n1 =
156503881374173899106040027210320626006530930815116631795516553916547375688556673985142242828597628615920973708595994675661662789752600109906259326160805121029243681236938272723595463141696217880136400102526509149966767717309801293569923237158596968679754520209177602882862180528522927242280121868961697240587
c1 =
778457304478982476832816099134231078039741924838797715386016566648152666554766952616954013371245538514040380284131564878345003064559091285634743825270728272882032759427199987196123463221966942639677691658071332886121935095232777955566588770461008663287891639229524839905122161995566925536054878241761125689651

n2 =
811767903948129438954176678224245038915381036612900677497468112441499272938807771403600643202454602366489650358459283710738177024118857784526124643798095463427793912529729517724613501628957072457149015941596656959113353794192041220905793823162933257702459236541137457227898063370534472564804125139395000655909
c2 =
407874861054070639330870597178271073295655401041548713389029773891369767064053212323564794615015075020723667207124492401853425282625784455322440983696547422848141750794119158481143278801448836205173367931653298932956857735156962602993084076125359920986051568222816877189044145334801497753299480858007260892841

n3 =
140612513823906625290578950857303904693579488575072876654320011261621692347864140784716666929156719735696270348892475443744858844360080415632704363751274666498790051438616664967359811895773995052063222050631573888071188619609300034534118393135291537302821893141204544943440866238800133993600817014789308510399
c3 =
100744134973371882529524399965586539315832009564780881084353677824875367744381226140488591354751113977457961062275480984708865578896869353244823264759044617432862876208706282555040444253921290103354489356742706959370396360754029015494871561563778937571686573716714202098622688982817598258563381656498389
```

```
39630

e = 3
```

Solution

`e` so small so I thought I could use to get 3 numbers and combined it to get flag but this was I got:

```
Bi....pT2p.@.'Xy51.. 3..d.8d&.fgS40...B.#%2r6.."(t.#D"I$.&8.V6..Y.P.p2X.(.'
IY..67PW#.is.b9.
q.7..u4.EvFS.6!qG`&3'f.g.`..%.5..q..U3..R.`.2..GR..P(XE.c..B$.
```

So I had to use other approach. At first, I checked whether each pair of these numbers had `gcd = 1` with Python:

```
def la_nguyen_to_cung_nhau(a, b, c):
    # Count GCD
    gcd_ab = math.gcd(a, b)
    gcd_abc = math.gcd(gcd_ab, c)
    return gcd_abc == 1

if la_nguyen_to_cung_nhau(c1, c2, c3):
    print("Correct")
else:
    print("Incorrect")
```

And result revealed that "Correct"
$\Longrightarrow$ From this point, I found I could use `CRT (Chinese Remainder Theorem)` : If `a,b,c` have `gcd(a,b,c) = 1`, then this system of residual equations has a unique solution modulo their product:

$\Longrightarrow$ Exist only 1 number from $\rightarrow$ that need to be found

According that, I got:

```
N=178640925972487154310997489019432159239959023145578726942731705176596326091
756652523872556350630121191049467600893941942173042988371031162669438419305
734610287041783008320823835397546099282726232795953849490675860255743431413 0
958157310583224713237293318566369040898788487966022114698274593911472572933 7
431869771147448401619420643790029176838877278355955881598378387755054490544 3
767694505870314577308437444494709622529775271960774117231383995822630973156 2
988750940806116363582751731727461017493745795643992439806678840104092904421 5
```

```
132296529041033207918267268256116576633837100116755908095372865748184100245618586040493454963432984865562287207389262025146848589339961924545240110542281653393521596538947273226727993074863578880293383136244107592073926459999021491279150081834445986276892428546646084563152278088196064644444715846440529472294680963557282533937900524624500153586045117695747463307066379995858022431922517365846 17
```

After that, found :

Next, compute the modulo inverse of in term of :

$$\rightarrow$$

At last:

Because, this challenge didnt have a big message so I could guest
Used a simple Python code for all steps:

```
from Crypto.Util.number import long_to_bytes, inverse

import math

from gmpy2 import iroot

n1 =
156503881374173899106040027210320626006530930815116631795516553916547375688556673985142242828597628615920973708595994675661662789752600109906259326160805121029243681236938272723595463141696217880136400102526509149966767717309801293569923237158596968679754520209177602882862180528522927242280121868961697240587

c1 =
778457304478982476832816099134231078039741924838797715386016566648152666554766952616954013371245538514040380284131564878345003064559091285634743825270728272882032759427199987196123463221966942639677691658071332886121935095232777955566588770461008663287891639229524839905122161995566925536054878241761125689 65

n2 =
811767903948129438954176678224245038915381036612900677497468112441499272938807714036006432024546023664896503584592837107381770241188577845261246437980954634277939125297295177246135016289570724571490159415966569591133537941920412209057938231629332577024592365411374572278980633705344725648041251393950006559 09
```

```
c2 =
407874861054070639330870597178271073295655401041548713389029773891369767064053212323564794615015075020723667207124492401853425282625784455322440983696547422848141750794119158481143278801448836205173367931653298932956857735156962602993084076125359920986051568222816877189044145334801497753299480858007260892840
n3 =
140612513823906625290578950857303904693579488575072876654320011261621692347864140784716666929156719735696270348892475443744858844360080415632704363751274666498790051438616664967359811895773995052063222050631573888071188619609300034534118393135291537302821893141204544943440866238800133993600817014789308510399

c3 =
100744134973371882529524399965586539315832009564780881084353677824875367744381226140488591354751113977457961062275480984708865578896869353244823264759044617432862876208706282555040444253921290103354489356742706959370396360754029015494871561563778937571686573716714202098622688982817598258563381656498389039630

e = 3

# Find gcd(a,b,c)
def la_nguyen_to_cung_nhau(a, b, c):
    # Tính GCD của ba sô´
    gcd_ab = math.gcd(a, b)
    gcd_abc = math.gcd(gcd_ab, c)
    return gcd_abc == 1

if la_nguyen_to_cung_nhau(c1, c2, c3):
    print("Correct")
else:
    print("Incorrect")                    #Result was correct

mul = n1 * n2 * n3        # Count N

M1, M2, M3 = 0, 0, 0

M1 = mul// n1             # Count M_i

M2 = mul// n2

M3 = mul// n3
```

```
y1,y2,y3 = 0,0,0         # Count y_i
y1 = inverse(M1, n1)
y2 = inverse(M2, n2)
y3 = inverse(M3, n3)


C = c1 * y1 * M1 + c2 * y2 * M2 + c3 * y3 * M3
C = C % mul                                      # Count C

# convert C to string to print easier
print(str(C))

# Giải mã C với e = 3
m, exact = iroot(C, 3)   # Find 3rd root
if exact:
    print("m =", m)      # Count m but m so big so I had to put its value it
after

m =
60433842571798516984027641963751231488962044650049325614160760426814530199862747686172874727584789847975622401694326338765578039636786277426075329564554
42

plaintext = int.to_bytes(m, (m.bit_length() + 7) // 8, "big")

print(plaintext)
```

The last part to print `plaintext`:
`int.to_byte`: Convert a integer number to bytes
`m.bit_length()+7 // 8`:
A way to count how many bits need to present a number `m`.
`+7` and `//8` to count how many bytes (1 byte = 8 bits)
Ex: A number have 100 bit need 100 + 7 // 8 byte
`big`: When present RSA, often use `big-endian`
Result:
b"scriptCTF{y0u_f0und_mr_yu's_s3cr3t_m3g_12a4e4}\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12\x12"

Flag: `scriptCTF{y0u_f0und_mr_yu's_s3cr3t_m3g_12a4e4}`

Div

Server's code:

```
import os
import decimal
decimal.getcontext().prec = 50

secret = int(os.urandom(16).hex(),16)
num = input('Enter a number: ')

if 'e' in num.lower():
    print("Nice try...")
    exit(0)

if len(num) >= 10:
    print('Number too long...')
    exit(0)

fl_num = decimal.Decimal(num)
div = secret / fl_num

if div == 0:
    print(open('flag.txt').read().strip())
else:
    print('Try again...')
```

Solution:

This was a very simple challenge because this server used `decimal.Decimal(num)` but `num` only used `input...` not `int(input())`

So that I could put: "Infinity" and got the flag

Because `decimal.Decimal` in Python support several special value in term of `IEEE 754` and `Decimal spec` :

```
decimal.Decimal("Infinity")
```

This can be understand like

Flag: `scriptCTF{70_1nf1n17y_4nd_b3y0nd_294da992e2e4}`

Off-by-one

Description:

i hid a qr inside a qr

Provided:

Challenge provided a `hidden.png` file which presented a QR code. Scanned it and got `Rick-roll`

Solution:

From the hint of this challenge, I thought I had to create new QR code from provided QR code

At first, I used `binwalk` to see whether this file had anything in it because it name was "`hidden`" and I got a `29` data file but could not do anything with it

After that, I found I could use a online tool `steghide` online to see detail of this `png` file and I found when turned to `Blue 0`, at the top of the image became torn so I thought this could be a valuable information

Next, I extracted file with `Blue 0` and at the `Hex` tab, copied a part of hex string:

```
000000000000000fe423f8416cd042e9c0ba175ae5d0ba28ae841519043faaafe00010000fbe
dd507ce16a80a9763e120e6a20ffd7e6842356b025f148610054e20a38ffa800734683fb6ab6
10425100bad1fb85d4638c2eb1d5210519710feaa058000000000000007f
```

I copied only a part of string because the rest was the background of QR image and if I copied all, this could make me difficult to create a new image at correct size

Note that length of string copied was even because hex presented at `0x` and two numbers after

After had a that hex string, I used Python to create new QR code from that:

```
from PIL import Image
import math

# Extracted hex data with trailing zeros and some f-characters:
hex_data =
"000000000000000fe423f8416cd042e9c0ba175ae5d0ba28ae841519043faaafe00010000fb
edd507ce16a80a9763e120e6a20ffd7e6842356b025f148610054e20a38ffa800734683fb6ab
610425100bad1fb85d4638c2eb1d5210519710feaa058000000000000007f"

# Convert hex string to binary string:
data = bytes.fromhex(hex_data) # Convert hex to bytes
bin_str = ''.join(format(byte, '08b') for byte in data) #Each byte to 8-bit
binary

# Trim trailing binary characters by 1 until perfect square:
```

```
while True:
    length = len(bin_str)
    size = int(math.isqrt(length)) # sqare root of length to find size
    if size * size == length:
        break
    bin_str = bin_str[:-1] #Delete last bit when not a perfect square
print(f"Final length: {len(bin_str)}, Square size: {size}x{size}")

# Build QR code, 1 is black and 0 is white:
img = Image.new("1", (size, size))
for i, bit in enumerate(bin_str):
    x = i % size
    y = i // size
    img.putpixel((x, y), 0 if bit == "1" else 1)

# Scale up for readability
img = img.resize((size*10, size*10), Image.NEAREST)
img.save("qr_flag.png")
img.show()
```

Flag: `scriptCTF{qrqrqrc0d3s}`

Plastic Shield

Description:

OPSec is useless unless you do it correctly.

Provided:

A ELF 64-bit LSB executable file

Solution

This is `main` function:

```
undefined8 main(void)

{
  size_t sVar1;
  undefined1 local_348 [256];
  undefined1 local_248 [16];
  undefined1 local_238 [32];
  char local_218 [64];
  undefined1 local_1d8 [64];
  undefined1 local_198;
```

```
  char local_189;
  byte local_188 [64];
  char local_148 [263];
  byte local_41;
  void *local_40;
  char *local_38;
  ulong local_30;
  size_t local_28;
  ulong local_20;
  ulong local_18;
  ulong local_10;
  printf("Please enter the password: ");
  __isoc99_scanf("%255s",local_148);
  local_28 = strlen(local_148);
  local_30 = (local_28 * 0x3c) / 100;
  local_189 = local_148[local_30];
  crypto_blake2b(local_188,0x40,&local_189,1);
  for (local_10 = 0; local_10 < 0x40; local_10 = local_10 + 1) {
    sprintf(local_218 + local_10 * 2,"%02x",(ulong)local_188[local_10]);
  }
  local_198 = 0;
  local_38 =
 "713d7f2c0f502f485a8af0c284bd3f1e7b03d27204a616a8340beaae23f130edf65401c1f9
9fe99f63486a385ccea217"
  ;
  hex_to_bytes(local_218,local_238,0x20);
  hex_to_bytes(local_1d8,local_248,0x10);
  sVar1 = strlen(local_38);
  local_18 = sVar1 >> 1;
  local_40 = malloc(local_18);
  hex_to_bytes(local_38,local_40,local_18);
  AES_init_ctx_iv(local_348,local_238,local_248);
  AES_CBC_decrypt_buffer(local_348,local_40,local_18);
  local_41 = *(byte *)((long)local_40 + (local_18 - 1));
  if ((local_41 < 0x11) && (local_41 != 0)) {
    local_18 = local_18 - local_41;
  }
  printf("Decrypted text: ");
  for (local_20 = 0; local_20 < local_18; local_20 = local_20 + 1) {
    putchar((uint)*(byte *)(local_20 + (long)local_40));
  }
  putchar(10);
  free(local_40);
  return 0;
}
```

Operation of this function:

1. `local_148` save user input
2. `local_28` save length of input
3. `local_30` save a value which equal to 60% of `local_28`
4. `local_189` save character which `local_148[local_30]`
5. `crypto_blake2b(local_188,0x40,&local_189,1)` : Hash the character at `local_189` to 64 byte with `Blake2b` and save at `local_188`
6. Convert that hashed string into 128 character in hex (0x...)
7. `local_198 = 0`
8. `local_38 = "713d7f2c0f502f485a8af0c284bd3f1e7b03d27204a616a8340beaae23f130edf65401c1f99fe99f63486a385ccea217"`
9. Generate `key` and `IV (initialization vector)`, `key` get from first 32 byte of `local_188` and `IV` get from next 16 byte after `key` which use for decode `local_38`
10. The rest of function operate for `blake2b` and print flag if correct password

This operation look very complex but it have a huge problem that it depend only on the character locate at position 60% length of input
$\Longrightarrow$ Therefore I can brute force from " " to "~" for get what character I need
Example: with a string 123456<"this position I can put all character for brute force">890

Python script:

```
from hashlib import blake2b

from Crypto.Cipher import AES

import binascii

# Ciphertext hex (copy từ code C)

ciphertext_hex =
"713d7f2c0f502f485a8af0c284bd3f1e7b03d27204a616a8340beaae23f130edf65401c1f99
fe99f63486a385ccea217"

def decrypt(password: str):
    # B1: Tính vị trí ký tự đặc biệt (36% độ dài password)
    idx = (len(password) * 0x3c) // 100
    special_char = password[idx].encode()

    # B2: Hash ký tự bằng blake2b -> 64 byte
    h = blake2b(special_char, digest_size=64).digest()
```

```
    # B3: Lấy key (32 byte) và iv (16 byte)
    key = h[:32]
    iv = h[32:48]

    # B4: Chuyển ciphertext hex thành bytes
    ciphertext = binascii.unhexlify(ciphertext_hex)

    # B5: AES-CBC giải mã
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted = cipher.decrypt(ciphertext)

    # B6: Bỏ padding PKCS#7
    pad_len = decrypted[-1]
    if 0 < pad_len <= 16:
        decrypted = decrypted[:-pad_len]

    return decrypted.decode(errors="ignore")

if __name__ == "__main__":
    pw = input("Please enter the password: ")
    result = decrypt(pw)
    print("Decrypted text:", result)
```

Flag: `scriptCTF{20_cau541i71e5_d3f3n5es_d0wn}`