Yêu cầu 2.1:



Giải thích: câu lệnh trả về một object (kiểu O) có str\_len=4, tiếp theo là object\_name=”User” và cuối cùng là object\_size=2 (bao gồm “name” và “isLoggedIn”), tiếp theo sau trong {} là các biến cũng như giá trị của biến theo format “kiểu”:”size của biến”:”giá trị của biến”

Text

Description automatically generated

Text

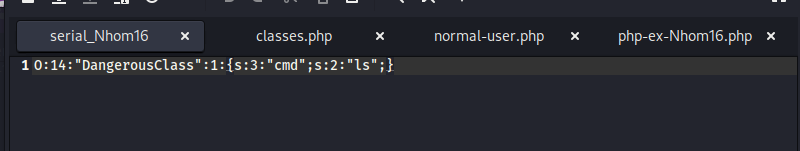
Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

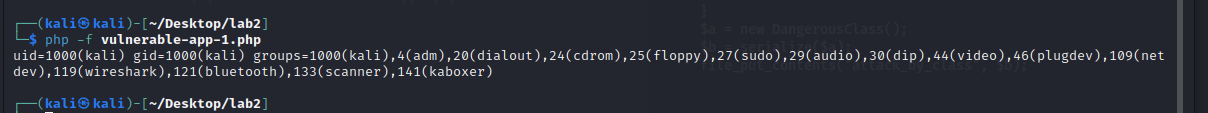


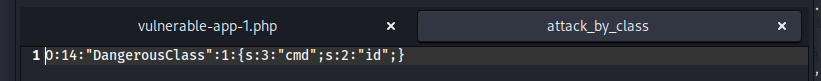
Yêu cầu 2.2:

Yêu cầu 2.3:

Text

Description automatically generated





B.2.2 Khai thác định dạng Java Serialiazition

Text

Description automatically generated

Text

Description automatically generated

Text

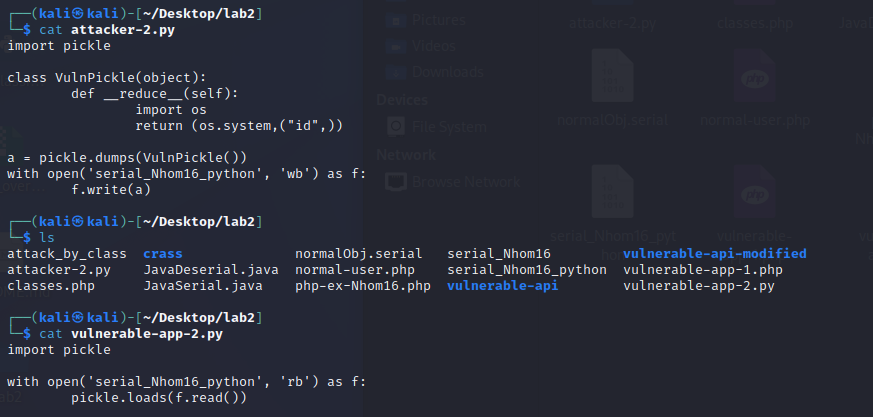
Description automatically generated

Yêu cầu 2.4:

- Ký tự rO0AB là ký tự mở đầu cho object được Serialize (Base64: rO0AB, Raw: 0xac, 0xed, 0x00, 0x05)

- file tấn công JavaSerial.java tiến hành serialize câu lệnh thực thi “ls” của os sau đó ghi vào file normalObj.serial dưới dạng một obj bình thường, sau đó khi file JavaDeserial.java thực thi, nó sẽ đọc file normalObj.serial để tiến hành deserialize obj có trong file và từ đó câu lệnh “ls” được thực thi mà không phải qua lớp kiểm tra nào của hệ điều hành

B.2.3 Khai thách định dạng Python Serialization



A screenshot of a computer

Description automatically generated with medium confidence

Yêu cầu 2.5 :

* Vì khi thực thi file attacker-2.py, tạo ra file serial\_Nhom16\_python chứa 1 object của lệnh id của os đã được serialize
* Sau đó file vulnerable-app-2.py đọc file serial\_Nhom16\_python chứa obj id sẽ không kiểm tra mã nguồn cũng như các chế độ security nên sẽ Deserialize luôn obj chứa lệnh id của os và thực thi nó nên sẽ trả ra kết quả như hình, bao gồm users và groups name và UID

Yêu cầu 2.6 :

Code tấn công :

A screenshot of a computer

Description automatically generated with medium confidence

* Class VulnPickle khi thực thi sẽ gọi os để thực thi lệnh nc 127.0.0.1 4016 để kết nối tới shell của attacker đang mở (reverse shell)
* Ta sẽ dùng hàm pickle.dumps() để tạo ra 1 obj được serialize
* Base64 là để convert obj thành base64
* Hàm decode() để convert từ byte sang string để gán vào trường hack của http request
* Dòng 16 là câu lệnh tạo POST request lên web server đang mở với trường hack được gửi kèm payload chứa class VulnPickle đã được tạo ở trên

Tiến hành tấn công :

* Text

  Description automatically generatedĐầu tiên ta sẽ mở port 4016 trên máy attacker để lắng nghe kết nối
* Tiếp theo ta sẽ cho chạy web server đã được dựng sẵn

Graphical user interface, text

Description automatically generated

* Sau đó ta sẽ chạy attack\_pickle.py. Và sau đó là kết quả đã tấn công được vào server bằng reverse shell

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence