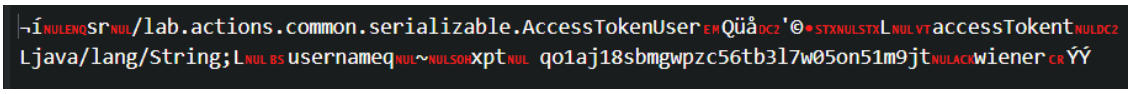
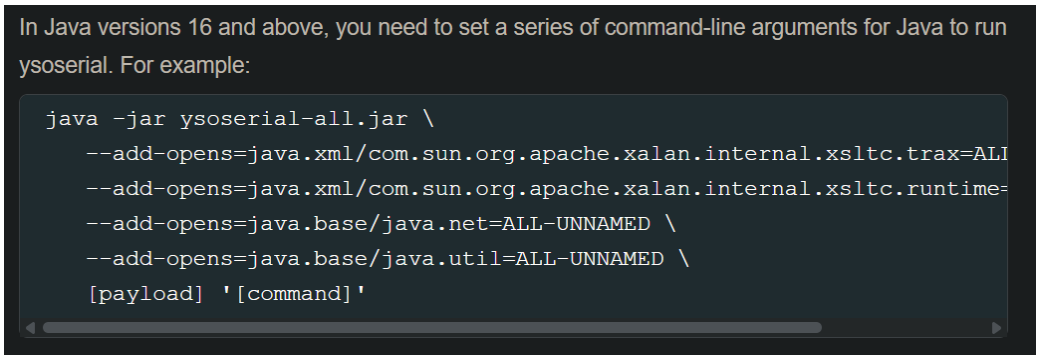
Exploiting Java deserialization with Apache Commons

1. This lab uses a serialization-based session mechanism and loads the Apache Commons Collections library. Although you don't have source code access, you can still exploit this lab using pre-built gadget chains. To solve the lab, use a third-party tool to generate a malicious serialized object containing a remote code execution payload. Then, pass this object into the website to delete the morale.txt file from Carlos's home directory.

2. I found the session in the request start with **rO0A** which means that this is base64 java serialized object. When I decode it, it is unreadable, but I noticed that it contains username and access token. So, I decided to use **ysoserial** tool to create a new serialized session.

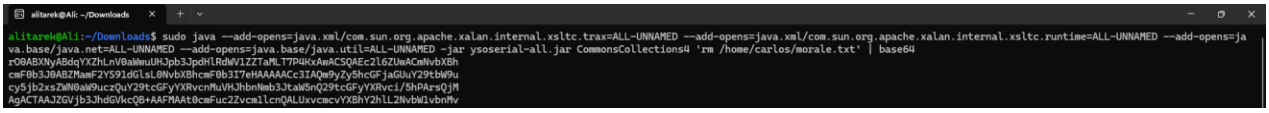


3. From lab's hint, I noticed that I should use a specific payload for Java version 16 and above.



4. I copied this and replaced the payload with **CommonsCollectionsX** and the command with 'rm /home/carlos/morale.txt'. Then I brute force **X** value till I tried **X=4**, the exploit ran successfully.

5. But there was an issue, when I run this command, it was didn't run because **– add-opens** must be before **-jar**. I replaced them and the command ran and I got the new session.



6. I copied this new session and replaced it with the old session in the request and sent. But it still didn't work.

7. When I encoded all characters to URL encode, the lab solved successfully but with **500 Internal Server Error** because the server didn't receive the access token. But it works.