

# Assignment 4.1

## Name: Session Attacks

### Developers:

- Anders Carlsson
- Oleksii Baranovskyi

### Performer:

- FB-01 Sakhnii Nazar

## Table of Contents

Task 1. Session Prediction/Fixation .....	1
---	---

## Task 1. Session Prediction/Fixation

**Purpose:** understand what is session prediction

### After the work, the student must

- know: what are session prediction and session fixation attack;
- be able to: recognize and analyze cookies and session options, exploit vulnerabilities.

### Tasks:

- analyze provided web application on virtual machine 192.168.56.6 and check parameters of its' sessions.

### Technical equipping of the workplace:

- Browser Developer Tools
- OWASP Burp Suite
- OWASP Zend Attack Proxy

### Solution:

Open the site in browser. Analyze HTTP-parameters for GET/POST requests. Use provided tools for detection and exploiting session vulnerabilities.

## TASK 1

For provided site, you need to answer: is session prediction vulnerability present? How did you find it? Prove it (screenshot). Explain why it is possible.

## Answer:

**Burp Suite Professional v2022.8 - Temporary Project - licensed to Uncia**

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn Hackvortor

Site map Scope Issue definitions

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

**Contents**

Host	Method	URL	Params	Status	Length	MIME type
http://192.168.56.6	GET	/		200	2956	HTML
http://192.168.56.6	GET	/?flush=false	✓	200	2920	HTML
http://192.168.56.6	GET	/?flush=true	✓	200	2920	HTML
http://192.168.56.6	GET	/data/bootstrap.min.js		200	37337	script

**Issues**

- Unencrypted communications
- Vulnerable JavaScript dependency (2)
- Duplicate cookies set**
- Cookie without HttpOnly flag set
- Frameable response (potential Clickjacking)

**Request** **Response**

Pretty Raw Hex Render Hackvortor

1 HTTP/1.1 200 OK  
2 Date: Wed, 10 May 2023 16:19:18 GMT  
3 Server: Apache/2.4.38 (Debian)  
4 Vary: Accept-Encoding  
5 Content-Length: 2764  
6 Connection: close  
7 Content-Type: text/html; charset=UTF-8  
8  
9 <!DOCTYPE html>  
10 <html lang="en">  
11 <head>  
12 <meta charset="utf-8">  
13 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
14 <meta name="viewport" content="width=device-width, initial-scale=1">  
15 <!-- The above 3 meta tags \*must\* come first in the head; any other head  
content must come \*after\* these tags -->  
16 <meta name="description" content="">

**Inspector**

**Duplicate cookies set**

Issue: Duplicate cookies set  
Severity: Information  
Confidence: Certain  
Host: http://192.168.56.6  
Path: /

**Issue detail**

The application attempted to set the following cookie to multiple values:

- user

**Send** **Cancel** **<** **>**

Target: http://192.168.56.6 HTTP/1


**Request**

Pretty Raw Hex **Render** Hackvortor

1 GET / HTTP/1.1  
2 Host: 192.168.56.6  
3 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:102.0) Gecko/20100101 Firefox/102.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Referer: http://192.168.56.6/  
8 Connection: close  
9 Cookie: user=5555555555  
10 Upgrade-Insecure-Requests: 1  
11  
12

**Response**

Pretty Raw Hex **Render** Hackvortor



She is waiting for you! But why other couldn't see her?

**Inspector**

Selection 15

**Selected text**

user=5555555555

**Decoded from:** URL encoding

user=5555555555

**Cancel** **Apply changes**

**Request Attributes** 2  
**Request Query Parameters** 0  
**Request Body Parameters** 0  
**Request Cookies** 1  
**Request Headers** 9  
**Response Headers** 6

Done 2 920 bytes | 3 millis

**Send** **Cancel** **<** **>**

Target: http://192.168.56.6 HTTP/1

**Request**

Pretty Raw Hex **Render** Hackvortor

1 GET / HTTP/1.1  
2 Host: 192.168.56.6  
3 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:102.0) Gecko/20100101 Firefox/102.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8  
5 Accept-Language: en-US,en;q=0.5  
6 Accept-Encoding: gzip, deflate  
7 Referer: http://192.168.56.6/  
8 Connection: close  
9 Cookie: user=-8888  
10 Upgrade-Insecure-Requests: 1  
11  
12

**Response**

Pretty Raw Hex **Render** Hackvortor



But other user can see hacker-woman!

**Inspector**

Selection 10

**Selected text**

user=-8888

**Decoded from:** URL encoding

user=-8888

**Cancel** **Apply changes**

**Request Attributes** 2  
**Request Query Parameters** 0  
**Request Body Parameters** 0  
**Request Cookies** 1  
**Request Headers** 9  
**Response Headers** 6

Done 2 956 bytes | 3 millis

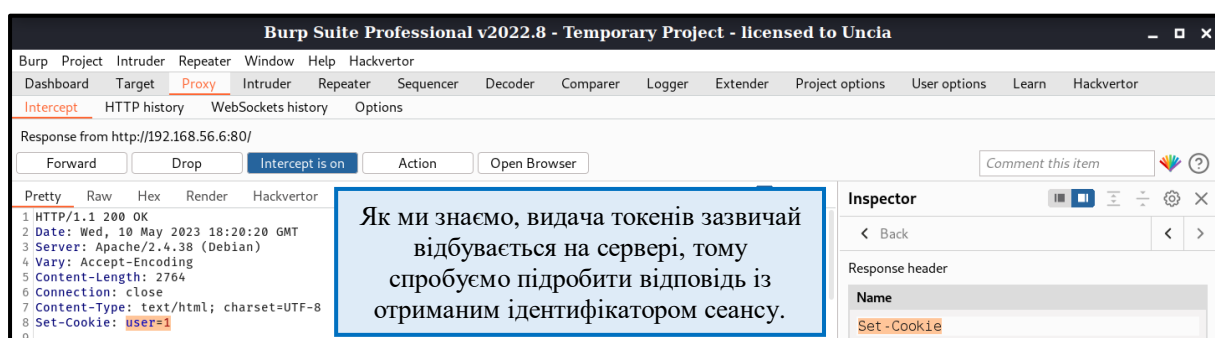
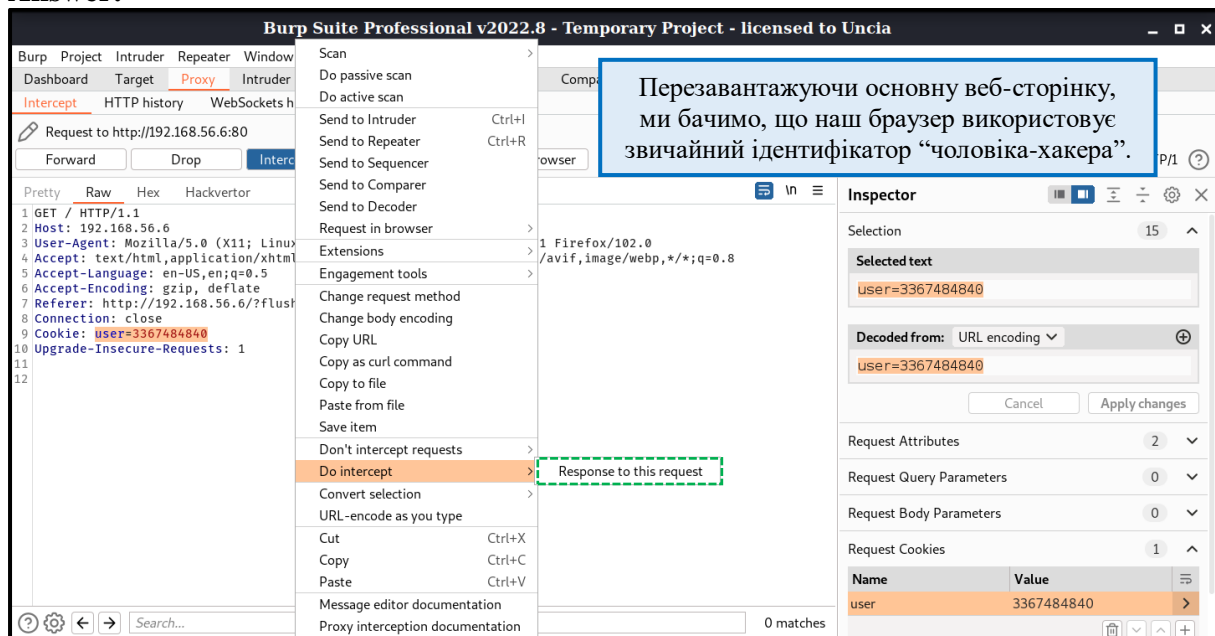
Хм, ну що тут можна сказати... Схоже, що токени сесій однозначно ідентифікують, якою є твоя роль на цьому веб-сайті (або ти бачиш зображення хакера-жінки, або тобі доступне лише фото хакера-чоловіка, який у стилі агітаційних плакатів армії США часів Першої та Другої світової війни запитує у тебе, чи ти готовий вступити у ряди передових хакерів). А так як моя відповідь: “Звісно я готовий! Тільки запропонуйте мені робоче місце, і я одразу приступлю до роботи.” – отже, мені не завдасть клопоту, відповісти, чому цей вид вразливості (**session prediction**) був можливий.

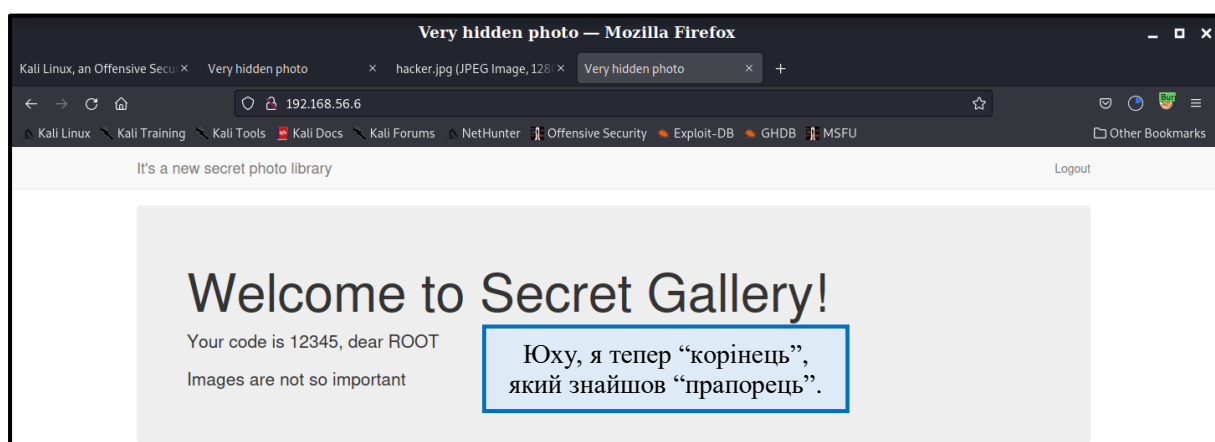
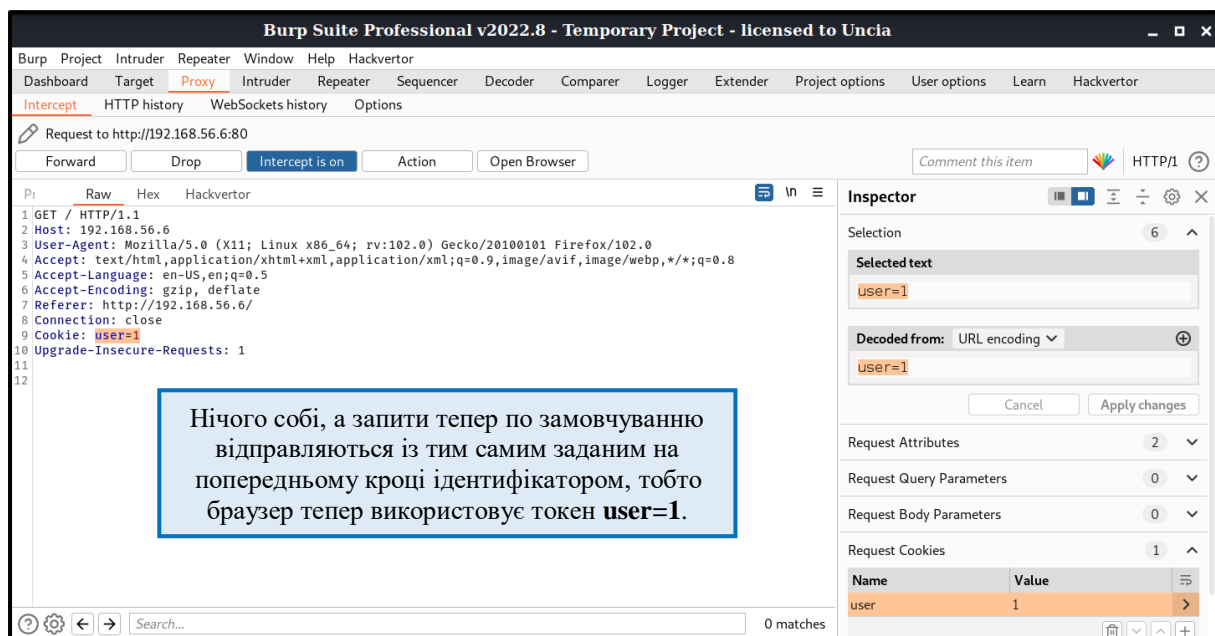
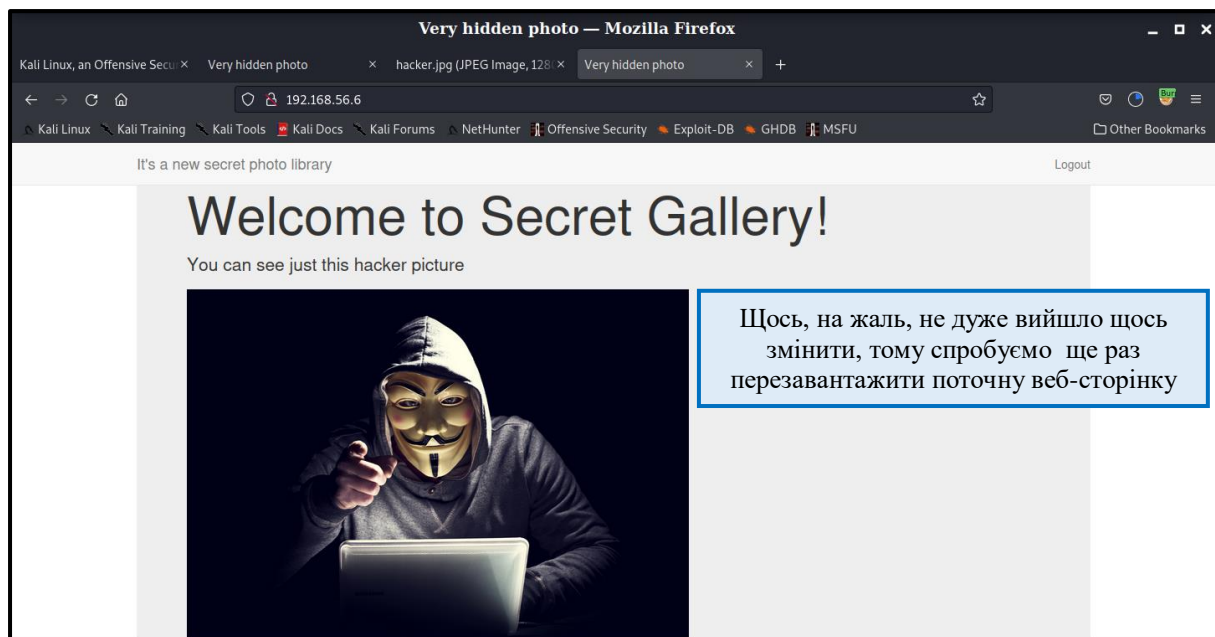
Отже, слухайте наступне. Ця вразливість існує, оскільки програма покладається на ідентифікатор сеансу або токен для зберігання додаткової інформації про користувача без належної перевірки чи захисту цих даних. Це дозволяє зловмисникам маніпулювати даними сеансу та потенційно отримати доступ до конфіденційної інформації чи функцій. Та й в цілому алгоритм генерації токенів був одразу визначений, так як він досить простий та передбачуваний. Як можна було помітити, усі цілі парні числа відповідають за роль чоловіка-хакера, а всі непарні відповідно за жінку-хакера. Тому та й таке...

## TASK 2

Is it possible to execute a session fixation attack? Prove it (screenshot). Explain why it is possible.

Answer:





Так би мовити, ось ми і скористалися даним типом вразливості (**session fixation**). Тобто ми напряму маніпулювали відповіддю сервера, перехоплюючи пакети, якими обмінюються клієнт і веб-додаток, для того, щоб встановити деякий параметр у **Set-Cookie**, що дало змогу отримати доступ до облікового запису привілейованого користувача. Отже, висновок такий, що відсутність захищеного з'єднання та слабкий контроль безпеки на стороні сервера, то є справжня біда для вашого веб-застосунку.