CSE 345/545: Foundations of Computer Security
Assignment: 2
Maximum marks: 100

Instructions:

- Follow the naming convention and submission instructions for every question carefully. Zip your submission files to a folder and name it **ASSIGNMENT2_<first_name>_<Roll no.>**
- **You only need to submit a single report containing submission for below questions. Report file name should follow the convention ASSIGNMENT2_<first_name>_<Roll no.>.pdf.**
- **Post your queries on Google Classroom. No Individual emails will be entertained.**
- **Strict plagiarism checks will be conducted for each question. The assignment must be done individually.**
- **Deadline for submission: 5th November 2023, 11:59 PM**

Q1. Phishing — 20 marks

1. Make a simple application using any framework you like (pygame, tkinter, React etc.) which resembles any service/application that you have used. Make this application look as authentic as possible. But this should be a phishing application/website. Embed a malicious link in one of the buttons which causes something which isn't supposed to happen (for eg shuts down the pc). Do not make the attack too serious that it toys around with the kernel. **[15]**
2. Give some ways by which we can prevent ourselves from being a victim of phishing **[5]**

   Deliverables :
   1. The application related files and the description and demonstration of the attack in the combined report
   2. The ways are also be written in the same report.

Q2. Burpsuite — 25 marks

Introduction:
Burp Suite is a powerful web application testing tool used by security professionals for penetration testing. In this assignment, you will explore the OWASP Juice Shop web application, identify vulnerabilities, and demonstrate your understanding of using Burp Suite.

Install the docker image for a vulnerable application server i.e. juiceshop (https://github.com/bkimminich/juice-shop).

Task 1: Proxy Configuration (5 marks)

a. Browser Proxy Configuration. **[2.5]**

   Configure your browser to use Burp Suite as a proxy.

Provide screenshots showing the configuration settings in both Burp Suite and the browser.

b. Intercepting and Modifying Traffic **[2.5]**

Explore the OWASP Juice Shop while intercepting the traffic through Burp Suite. Provide screenshots of intercepted requests and responses that demonstrate your ability to manipulate requests.

Deliverable: Report that includes screenshots and step-by-step instructions on how to configure your browser to use Burp Suite as a proxy. Provide explanations for each step.

Task 2: Impersonating Users (5 marks)

a. "Post some feedback in another user's name" **[5]**

Explain how you posted feedback in another user's name.
Provide details of the user you impersonated, including their username.
Describe the steps you followed to successfully impersonate the user.
Include all screenshots that support your explanation.

Deliverable: Screenshots and a detailed step-by-step guide on how to post feedback in another user's name. Include explanations and any relevant information to support your explanation.

Task 3: Exploiting SQL Injection and Cracking MD5 Hash (15 marks)

a. "Retrieve a list of all user credentials via SQL Injection" **[7.5]**

Describe how you retrieved a list of all user credentials via SQL injection.
Include screenshots of your SQL injection payload and the results.

b. "Log in with Jim's user account" **[7.5]**

Explain how you cracked Jim's MD5 hash and successfully logged in.
Provide screenshots and the clear text password you obtained.

Deliverable: For "Retrieve a list of all user credentials via SQL Injection," provide a detailed guide on how you retrieved user credentials via SQL injection, including screenshots and the SQL injection payload used.
For "Log in with Jim's user account," explain how you cracked Jim's MD5 hash and successfully logged in. Include screenshots and the clear text password obtained.

Q3 BlockChain **30 marks**

A "smart contract" is a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Additionally, you can also use blockchain in your group projects for high security tasks like payments etc.

a. Write a smart contract in Solidity to mint NFTs using the ERC-721 Standard of tokens. Deploy the smart contract on the Goerli Testnet. Use your Metamask wallet address to create the contract on the Testnet. An example deployed contract can be found here.
Deliverable: **The Solidity smart contract named "NFT_FCS.sol".** The code should be *well-documented* with comments about each function and class variables. **[5]**

b. Mint an NFT of an image unique to you using the deployed contract. For example, it can be your passport size photo or a photo of an item only you own. An example minted NFT transaction can be found here. **[15]**

The NFT metadata should have the following JSON format:

```
{
  "name": "<image name>",
  "image": "<IPFS URI of the unique image>",
  "description": "<a one-line description of the image>",
  "properties": {
    ... (optional)
  }
}
```

Deliverable: **A JSON file named "nft-details.json" with the following format:**

```
{
  "metamask_public_address": "<your metamask wallet public address>",
  "contract_address": "<address of the deployed smart contract>",
  "nft_transaction_hash": "<hash of the transaction that minted the NFT>",
  "image_ipfs_cid": "<the IPFS content identifier of the unique image>",
  "metadata_ipfs_cid": "<the IPFS content identifier of the metadata of the
image>"
  }
```

c) Extend your above NFT contract by implementing the following features **[10]** :

i. transferNFT: Create a function that enables NFT owners to transfer their NFTs to another Ethereum address. **[3]**

ii. tradeNFT: Implement a function that allows two users to trade NFTs with each other. Each user should specify the NFT they want to trade, and the trade should be executed if both users are the rightful owners of the specified NFTs and have the necessary permissions. **[7]**

Deliverable: The Solidity smart contract named“Tradable_NFT_FCS.sol”

Q4 WebGoat - Pentesting Practice **25 marks**

WebGoat is a deliberately insecure web application maintained by OWASP. It is designed to help you practice testing vulnerabilities commonly found in web-based applications.

You can use docker to run Webgoat: https://github.com/WebGoat/WebGoat **(make use of a VM preferably, do not run WebGoat on your local machine with internet connected)**

Complete the **Broken Access Control(A1) , Injection (A3), Identity & Auth Failure(A7), Software and Data Integrity(A8)**, **Request Forgeries(A10)** Lessons along with their associated assignments. **(5x5 = 25)**

Deliverable: Provide screenshots & explanation of the steps you performed for the completion of assignments(**screenshots must contain date and time of your system**) in the report.