CSE 345/545: Foundations to Computer Security

Assignment: 2

Maximum marks: 100

Instructions:

- Follow the name convention and submission instructions strictly and carefully. The questions may be evaluated against a script and marks will be deducted for not following the naming conventions.
- Name the report: <Roll no.>_A2.pdf.
- The report, JSON/code files must be zipped and submitted, eg. <Roll no.>_A2.zip.
- Post your queries on Google Classroom.
- **Strict plagiarism checks will be conducted for each question. The assignment must be done individually.**
- Deadline for submission: 21th November 2022, 11:59 PM.

---

1. **Metasploitable** **40 marks**

Download '**metaspoiltable.zip**' (attached with the HW on classroom), install VM and bring the system up.

Default username: msfadmin and password: msfadmin.

Attack Machine: You can use Kali Linux or Ubuntu as the attacking machine. Kali Linux comes with a suite of applications pre-installed. Unless specified, you will perform the following exercise on the attacking machine.

Report format:

1. Background of the attack (3 bullet points on why / how / outcome).
2. Steps followed to perform the attack.
3. Appropriate screenshots for each command/attack.
4. Other deliverables are specific to the questions.

Problems:

a. Use Nmap to identify the OS version of the metaspoiltable system. [5 marks]

b. List the open ports on the metaspoiltable system. What commands did you use? What are the ports used for by default? What applications did you find running on the open ports? [10 marks]

c. Metasploitable contains a backdoor on its FTP server. Exploit the same and report the following:
   i. What tool(s) did you use? [5 marks]
   ii. What command(s) did you execute? [10 marks]
   iii. What is the outcome of the exploit? [5 marks]

d. Metaspoiltable has Mutillidae running on the VM. Mutillidae contains the top-10 vulnerabilities on OWASP. You are required to exploit the "Add blog for Anonymous" vulnerability on the "Cross Site Request Forgery (CSRF) page." [5 marks]

2. WebGoat **10 marks**

WebGoat is a deliberately insecure application that allows interested developers to test vulnerabilities commonly found in web-based applications that use shared and popular open-source components.

You can use docker to run Webgoat: https://github.com/WebGoat/WebGoat.

The following 5 short lessons include web application vulnerabilities and attacks. [2x5=10]

1. HTTP Proxies
2. Developer Tools
3. Crypto Basics
4. Authentication Bypasses
5. Insecure Login

Share screenshots of each completion and explain your methodology to exploit the vulnerabilities.

3. BurpSuite **25 marks**

Burp Suite is a set of tools used for penetration testing of web applications. Some key features include Proxy, Target, Repeater, and Intruder.

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

a. Configure your browser to use burpsuite as a proxy. Explore the OWASP Juice Shop and show the intercepted traffic in the burp suite. [5]

b. Give the customer feedback with an impossible rating of 0. [5]

c. Use your knowledge of SQL injection to achieve the following
   i. Get admin access to the web portal. [10]
   ii. Get the credentials of all registered users in the portal. [5]

4. Blockchain and Cryptocurrency 25 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.

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. [8]

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. [17]
   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>"
}
```