UNIVERSITY OF Ottawa Faculty of Engineering and Computer Science
CSI 4139-CEG 4399 - SEC 5100 – Fall 2024
Assignment 1

Problem 1-1. Password Cracker (10 %)

You're working for the NSA and have been assigned the task of breaking into the most top-secret government agencies. To do so, you must crack as many passwords that have been hashed using the MD5 algorithm as you can. The hashes have been included in a separate file (hashes.txt) and have been divided into weak, moderate, and strong categories. Try to break as many of them as you can! Please submit three Python lists of cracked passwords for each category. Leave passwords that you are unable to break as an empty string.

For example:
Weak:
1. 1f3870be274f6c49b3e31a0c6728957f
2. fe01d67a002dfa0f3ac084298142eccd

Answer: # Weak:
[' apple',"]

Problem 1-2. Enhancing a File Sharing Service with Attribute-Based Access Control (ABAC) (10 %)

A file-sharing service similar to Dropbox or Google Drive wishes to implement Attribute-Based Access Control (ABAC) to manage access to files and folders based on user attributes, resource attributes, and environmental conditions.

- **Design the ABAC model**: Outline the attributes you would use for both users and resources.
- **Policy examples**: Provide examples of access control policies that could be implemented using ABAC.
- **Security benefits**: Discuss the advantages of ABAC over role-based access control (RBAC) in the context of a file-sharing service.

Your submission should include policy pseudo-code and an explanation of how ABAC would handle dynamic access scenarios, such as access requests based on the time of day or the sensitivity of the document.

Problem 1-3. Chosen-Ciphertext Attack (10%)

The basic RSA cryptosystem is vulnerable to Chosen-Ciphertext Attack. This means if we have a ciphertext in hand, say c1, corresponding to an unknown message, M1, and then submit a chosen ciphertext, say c2, and receive its corresponding message, M2, then we can recover the unknown message, M1. Clearly shows how we can do this attack. Then, briefly explain how we can prevent this type of attack.

Note: The assumption is that the attacker knows c1, and can choose c2, and receive M2, such that E(M2) = c2. The attacker also knows the public key (e,n), but has no information about the private key d. Now, we would like to know how the attacker, having this info, can find M1.

Problem 1.4. Capture the flag (70%)

Goal: This problem introduces the concept of a CTF through the form of web vulnerabilities. Most of the vulnerabilities come from the OWASP top 10. OWASP. which stands for Open Web Application
Security Project is a project dedicated to making the web more secure. There are 7 vulnerabilities in this platform based on the OWASP top 10. Furthermore, vulnerabilities like the ones here have all been found in major production websites at some point.
Setup:
We are using docker. The following steps will have the exercise up and running in docker in no time.

1. Run the command:

```
docker load < lab4.tar
```

to load the image into docker.

2. Run the command:

```
docker-compose up -d
```

(note this must be run from the directory containing *docker-compose.yal*)

3. The lab should now be accessible from http://localhost:1337/

If you are unable to access the lab you may be running an older version of docker which runs in a VM with a separate IP. To get that IP run the command:

```
docker-machine ip
```

and then connect to http://<ip here>:1337/

When you are done with the lab you will need to kill the docker containers. This can be done using:

```
docker compose down
```

Note: This will wipe the database. If you do not want to wipe the database you can use:

```
docker stop lab4_web lab4_db
```

WARNING the following commands will remove everything you have in docker.

Finally when you're done with everything you can use:

```
docker volume prune
```

to completely wipe the docker environment. Alternatively, you can remove the images for the lab manually.

Questions:

Most of the challenges in this exercise have a flag associated with them. A flag is of the format "*flag{some_text_here}*". The idea comes from the concept of a cyber security CTF in which participants hunt for vulnerabilities revealing flags. These vulnerabilities typically reflect real-world security issues.

Question 1

Register for an account. You may need to use SQL injection to help you out here. You will find a flag once your SQL injection is successful

Question 2

Create a URL so that the user is redirected after logging in. There is no flag for this challenge. Just record the URL that you used.

Question 3

Find a stored cross-site scripting vulnerability and have an alert pop up on the dashboard page. Once successful, a second alert will pop up.

Question 4

A piece of information is exposed on the employee dashboard. Find it and the flag that comes with it.

Question 5

Based on the information exposed in question 4, are you able to find a file on the system that should not be accessible?

Question 6

Using the result of question 5, you need to **figure out** the password of **Fred**. This is another form of sensitive data exposure.

Question 7

Find a way to **change** the password of the **admin** account. It will be more difficult than figuring out Fred's one... There might be a way through the messaging system. After all, the admin's assistant does automatically read the messages.

After completion, submit your answers as a single pdf file on Brightspace.