Login Authentication Lab

This is an **individual project**. You may **not** work in groups.

The code and other answers you submit must be entirely your work. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside the class. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

Solutions must be submitted electronically via gradescope. The whole project is due at 11:59pm on **Friday, October 31**. Your submission MUST include the following information:

1. List of people you discussed the project with
2. List of references used (online material, course nodes, textbooks, wikipedia, etc.)

If any of this information is missing, at least 20% of the points for the assignment will automatically be deducted from your assignment. See also discussion on plagiarism and the collaboration policy on the course syllabus.

This lab will be administrated by Jonathan Xu.

Introduction

In this project, your objective is to successfully login to a website that we will provide to you using the credentials for *admin*. You may use any attacks or exploit vectors that we have discussed so far. The project consists of a series of challenges that will get progressively harder and might require a combination of techniques in order to achieve the objective. Your solution to each challenge should be a written report of what you did in order to hack into the *admin* account and any scripts that you have used/written in order to do so. When it comes to writing helper scripts that you will almost certainly need, **you should stick with Python for all of your scripting**.

Common Guidelines

Similar to the Bungle lab, we will provide you with the target website in the form of a docker container that you will run locally. Each challenge will be a separate docker container. Every challenge will contain a login page with ***username*, *name*, and *password* fields**. To complete a challenge, you must be able to obtain a password for the *admin* account and successfully log in using *admin* as the username and *the obtained password* as the password. In some challenges, there will be other accounts co-existing with the *admin* one as well. Hacking into them does not mean the completion of a challenge. **Your only goal is to log in under the *admin* username**. Once this is done, you will see a green screen saying "Access Granted" with a **hash underneath it** (you will see the same screen when you successfully log in using other valid credentials as well, however, as stated before, you are required to hack into the *admin* account).

There is **one more important thing**. In most of the challenges (unless stated otherwise), each time you re-run the Docker container the *admin* password will change. It will still abide by the restrictions set up by each challenge, however, the actual password will be selected randomly based on the provided scenario. For example, if a hypothetical challenge provides you information that the *admin* password is a numerical value between 1 and 9, after each restart of the docker container for this challenge, a random value in this region will be selected as the password.

Therefore, **you should NOT just give the *admin* password as your answer**. **Your answer should be a reproducible list of instructions and scripts that will allow you to log into the *admin* account every time you restart the docker container**.

In addition, you should **submit a hashes.txt file containing all the hashes obtained when the access is granted using your full name (first and last, all lowercase)** (Make sure to have the *name* field filled in the login with **your actual name**). Make sure **you also list the exact input you put in the field at the top of your** `hashes.txt`**.**

At this point, you should be already proficient with using Docker, scripting, and browser tools. Please refer to the Login Lab FAQ or come to office hours if you have any other questions/problems.

Important Notice

This project as all other ones in this class asks you to develop attacks and test them, with our permission, against a target website that we are providing for this purpose. Attempting the same kinds of attacks against other websites without authorization is prohibited by law and university policies and may result in *fines, expulsion, and jail time*. **You must not attack any website without authorization!** Per the course ethics policy, you are required to respect the privacy and property rights of others at all times, *or else you will fail the course.* See the “Ethics, Law, and University Policies” section on the course website.

Challenge 0: Getting Used to The Framework

This challenge is just a demo. **You are not required to submit anything for it**.

To get started run the following command in your terminal:

```
docker run --name challenge0 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge0
```

Now, if you open your browser and navigate to `localhost:8080`, you should be able to see the login page for Challenge 0.

Scenario

Through your sources you have been informed that the admin only uses one of five very simple passwords:

- `forest`
- `desert`
- `ocean`
- `mountain`
- `jungle`

Use this information to gain access to admin's account.

Hints

- Restart the docker container and check if the password changes. There is a 20% chance that this will happen.
- Before you move onto a new challenge, terminate this docker container. If you want to run multiple challenges at once, make sure you properly configure port-forwarding when you run the docker container.

Challenge 1: Dictionary Attack

To get started run the following command in your terminal:

```
docker run --name challenge1 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge1
```

Scenario

You have been informed that the person behind the admin account does not have a good imagination when it comes to choosing a secure password. Therefore, each time he chooses a password, it always falls into the 100 most commonly used passwords in the world. You can obtain this list here: password list.

Use this information to gain access to the admin's account.

Hints

- Analyze using browser tools what happens after you click *Login*.
- Solve this challenge with a script (I recommend you to look into Python's requests module).
- When making requests, make sure to include all fields.

Submission

Submit a `.pdf` file with a step-by-step explanation of your attack and any scripts you used. Attach your script file separately, make sure the name of the script file makes sense, and you properly reference it in the `.pdf`. Make sure to add your hash to a new line in the hashes.txt file.

Example: `challenge1.pdf`, `ch1.py`, and `hashes.txt`.

Challenge 2: Brute-force

To get started run the following command in your terminal:

```
docker run --name challenge2 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge2
```

Scenario

You know that the millennial behind the admin account always uses their first name and birthday as the password. You don't know who exactly operates the account; however, you managed to pinpoint 4 people. Their names are: Alice, Bob, Carol, and Eve.

Use this information to gain access to admin's account.

Hints

- You can assume the following about the password:
  - the person's name is in lower case
  - You can assume the password is in this format `"[name]-[day]-[month]-[year]"` with no white-space and all letters are *lower-case* (also include dashes). Also, if day or month is single digit, you can assume that "0" is prepended to the number.
- Again, keep in mind that the password changes every time you restart the Docker container. You should provide a unified set of instructions and scripts as your solution.
- You might also want to look into multi-threading to reduce your scripts' runtime.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. Make sure to add your hash to a new line in the `hashes.txt` file.

Example: `challenge2.pdf`, `ch2.py`, and `hashes.txt`.

Challenge 3: Hashing

To get started run the following command in your terminal:

```
docker run --name challenge3 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge3
```

Challenge 3 introduces the first update to the framework. **In addition to the login page, there is now a search page that allows you to search for all registered users by name**. You should navigate to `/search` to use it.

Scenario

You have investigated the programmer who designed the search page and discovered that he is completely terrible. He does not adhere to any security practices except hashing the passwords using the plain `md5` hash when storing them in a database. You also know that he exclusively uses sqlite for the database.

Use this information to gain access to admin's account.

Hints

- MD5 is a cryptographic hashing algorithm first designed in 1991. It was completely broken in 2004. You should Google "reversing MD5 hashes" and see if you can find anything that helps you solve this problem.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the scripts are short, you can include them inside your `.pdf`. Make sure to add your hash to a new line in the `hashes.txt` file. If you used any of online tools, please reference them.

Example: `challenge3.pdf`, `ch3.py`, and `hashes.txt`.

Important Notice

This is the only challenge where the password is fixed (e.g. it does not change when you restart the docker container). Once you figure the password out, **you are NOT allowed to share it with other classmates**. If we see you doing this, you will receive **zero** on the assignment!

Challenge 4: Collisions

To get started run the following command in your terminal:

```
docker run --name challenge4 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge4
```

Challenge 4 uses the same page structure as previous challenge. It means the `/search` page is available to you.

Scenario

The programmer who designed the search page decided that the problem with the system was that it used the `md5` hash function, which he now realizes is broken. Since he hasn't learned about rolling his own crypto, he decided to come up with his own secure hash function. You have learned that this function takes its input and truncates it to the first 48 bits, then XORs each byte with the hexadecimal value `0xa5`. This is stored in the database as a hexadecimal string.

Use this information to gain access to admin's account.

Hints

- Look up other cryptographic hash functions to see what properties they have that this one might be missing.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the scripts are short, you can include them inside your `.pdf`. Make sure to add your hash to a new line in the `hashes.txt` file. If you used any of online tools, please reference them.

Example: Example: `challenge4.pdf`, `ch4.py`, and `hashes.txt`.

Challenge 5: Salted SHA256

To get started run the following command in your terminal:

```
docker run --name challenge5 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge5
```

Challenge 5 uses the same page structure as previous challenge. It means the `/search` page is available to you.

Scenario

After suffering a fiasco in the previous challenge, another programmer decided (again) that it was solely the hashing problem that allowed the hack to take place. Therefore, she updated the hashing mechanism. Now, instead of plain hashing with MD5 or a homemade function, she decided to use salted SHA256. For some unknown reason, she also asked every user to change their password to a five digit numerical value.

You were also told by some insiders that due to a severe case of the Dunning-Kruger effect, the programmer thought that salting with a random lowercase two-letter string would be enough to keep the passwords secure. She believes that because each salt is unique, no attacker will be able to crack all of the passwords efficiently.

Use this information to gain access to admin's account.

Hints

- You can assume the following about the password:
    - The password is stored as a SHA256 hash of a UTF-8 encoded string
- Be warned that the `/login` and `/search` pages are rate-limited.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the scripts are short, you can include them inside your `.pdf`. Make sure to add your hash to a new line in the hashes.txt file.

Example: `challenge5.pdf`, `ch5.py`, and `hashes.txt`.

Challenge 6: Backdoor Backfired

To get started run the following command in your terminal:

```
docker run --name challenge6 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge6
```

Challenge 6 introduces another update to the framework. **In addition to the login and search pages, there is now a registration page that allows you to create an account**. You should navigate to `/register` to use it.

Scenario

After multiple hacks, the website owners decided to redesign the password storage mechanism. However, they left a backdoor in the password encryption to have access to plain-text passwords on demand. After talking to some people that operate the website, you know that the passwords are not hashed anymore but instead they are stored in an encrypted version. The encryption they decided to use is a Vigenère cipher with the website's internal secret key.

Use this information to gain access to admin's account.

Hints

- You can assume the following about the password:
  - The password is stored as an ASCII string after applying the Vigenère cipher using the password and secret key
  - The secret key is truncated to the length of the password
  - All characters in the secret key and encrypted passwords are stored as uppercase letters

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the scripts are short, you can include them inside your `.pdf`. Make sure to add your hash to a new line in the hashes.txt file.

Example: `challenge6.pdf`, `ch6.py`, and `hashes.txt`.

Challenge 7: Proper SHA256, but Still Vulnerable

To get started run the following command in your terminal:

```
docker run --name challenge7 -d -p 8080:8080 jonxuu/cs357_login_lab:challenge7
```

Challenge 7 introduces another update to the framework. **In addition to the login, search and register pages, there is now a profile page that allows you to view information about users**. You should navigate to `/profile` to use it.

Scenario

The website owners finally decided to change the password storage to using a proper SHA256. They decided not to use salting because they reasoned that SHA256 is a secure cryptographic hash. Of course, they made sure that everyone changed their passwords to adhere to modern security guidelines.

Use this information to gain access to admin's account.

Hints

- Your options now are either to spend an indefinite amount of time trying to reverse SHA256 hash or find some other way to log in as admin successfully.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the scripts are short, you can include them inside your `.pdf`. Make sure to add your hash to a new line in the hashes.txt file.

Example: `challenge7.pdf`, `ch7.py`, and `hashes.txt`.