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 **Nov 3 2023**. 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 Jinquan Pan and Ivan Izhbirdeev.

Introduction

In this project, your objective is to successfully login into a website that we will provide to you using the credentials for *admin*. You may any attacks or exploit vectors that we have discussed so far. The project consists of series of challenges that will get progressively harder and would require 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 are free to use any programming language that you are comfortable with. However, if you are unsure which one to choose, **we strongly recommend that you stick with Python for all of your scripting**.

Common Guidelines

Similar to the Bungle lab we will provide you the target website in a form of a docker container that you would have to run locally. Each challenge will be a separate docker container. Every challenge will contain a login page with *username* 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 a completion of a challenge. **Your only goal is to log in under *admin* username**. Once this is done, you will see a green screen saying "Access Granted" (you will see the same screen when you successfully login using other valid credentials as well, however, as stated before you are required to hack into *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 to 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 obtain the correct password (or log into *admin* account through other means) every time you restart the docker container**.

At this point, you should be already proficient with using Docker, scripting, and browser tools. Please refer to FAQ's for the Bungle lab or come to office hours if you have any 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 -d -p 8080:8080 izhbi/projectx: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 three very simple passwords:

- `monkey`
- `apple`
- `banana`

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

Hints

- Restart the docker container and check if the password changes. There is a 66% 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: 100 Most Common Passwords

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx: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 100 most commonly used passwords in the world. You can obtain this list here: password list.

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

Hints

- Analyze using browser tools what happens after you click *Login*.
- Solve this challenge with a script.

Submission

Submit a `.pdf` file with step-by-step explanation of your attack as well as any scripts you used. If the script is short, you can include it inside your `.pdf`. You can also attach a script file separetly, make sure the name of the script file makes sense and you properly reference it in the `.pdf`.

Example: `challenge1.pdf` and `ch1.py`.

Challenge 2: Brute-force

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx:challenge2
```

Scenario

You know that the person behind the admin account always uses their first name and month+year of their birthday as the password. You don't know who exactly operates the account, however, you managed to pinpoint 4 people. Their names are: Mike, Jane, John, Mark.

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

Hints

- You can assume the password is in this format `"[name][month][year]"` with no white-space and all letters are *lower-case*. Also, if month is single digit, you can assume that "0" is prepended to the number.
- Keep in mind that the password changes every time you restart docker container. So you should provide a unified set of instructions and scripts as your solution.

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`.

Example: `challenge2.pdf` and `ch2.py`.

Challenge 3: Hashing

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx:challenge3
```

Challenge 3 introduces 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`. If you used any of online tools, please reference them.

Example: `challenge3.pdf` and `ch3.py`.

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: Salted SHA256

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx:challenge4
```

Challenge 4 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, the programmer decided that it was solely the MD5 hashing problem that allowed the hack to take place. Therefore, he updated the hashing mechanism. Now, instead of plain MD5 hashing he decided to use salted SHA256. For some unknown reason, he 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 complete lack of competency, the programmer couldn't come up with a better salting mechanism than just prepending the month+year of the website creation date to the actual password. After some online research, you know for sure that the website didn't exist before 2019, however, you were unable to find the exact date of its creation.

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

Hints

- You can assume the password is stored as a SHA256 hash of a UTF-8 encoded string `"[salt][password]"`.
- Try using similar exploit as in previous challenge to get the password hash.

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`.

Example: `challenge4.pdf` and `ch4.py`.

Challenge 5: Backdoor Backfired

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx:challenge5
```

Challenge 5 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 just a simple XOR of the password and the website's internal secret key.

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

Hints

- Think about how the registration page can be helpful.
- Search page exploit should be helpful here as well.
- You can assume that the password is stored as a character-wise XOR between each character of the password as an ASCII value and each character of the secret key which is truncated to the length of the password as an ASCII value. After XORing each character this way, the result is appended to the encrypted version as a hex number (if resulting hex number is single digit, "0" is prepended to it).

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`.

Example: `challenge5.pdf` and `ch5.py`.

Challenge 6: Proper SHA256, but Still Vulnerable

To get started run the following command in your terminal:

```
docker run -d -p 8080:8080 izhbi/projectx:challenge6
```

Challenge 6 uses the same page structure as previous challenge. It means the `/search` page and `/register` pages are available to you.

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 spend an indefinite amount of time trying to reverse SHA256 hash or find some other way to successfully log in as admin.

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`.

Example: `challenge6.pdf` and `ch6.py`.