CS3235 Week 9 Demo

CS3235 Teaching Team

March 2022

This demo lets you explore how cookies are passed between a browser and a webserver and in what cases HTTP requests are allowed and in what cases they are considered risky and disallowed. It also comes with a simple example of a CSRF attack and the CSRF token defence.

To get the most out of this demo, you are encouraged to

- Use the "web developer tools" of your web browser to inspect each HTTP request and response (especially the headers), the output to the console, as well as the cookies stored by the browser. On both Firefox and Chrome, you can open it by pressing F12.
- Click around on the web pages included in this demo.
- Read the web page source code (in very simple HTML, PHP and JavaScript).

Setup

1. Set up the "lab environment" (see the released instructions).
2. Download `week9-demo.zip` and unzip it.
3. In your shell, execute `seclab-setup-demo`. This sets up a few domain names in your `/etc/hosts` and might prompt you for your password.
4. In your shell, execute `seclab-serve-demo <folder>` where `<folder>` is the folder you just unzipped the demo files to. Then open your browser, go to `http://localhost/`. A web page should show up. Open the developer tools, go to the "network" tab and disable caching there. Linux users only: if you prefer to use the Firefox bundled in the Docker image, you can use `seclab-serve-browse <folder>` instead. A Firefox window will show up. You don't need to use `seclab-setup-demo` or `seclab-clean-demo` either if you choose this option.
5. You can click around now. Enjoy!
6. You can use `seclab-clean-demo` to clean up the installed entries in `/etc/hosts` after you've had enough of this demo.

A Guided Tour

- `http://demo.org/origin.php`: you can find different types of cross-origin requests (subresources, XHRs, links, forms, iframes) and observe when each of them is allowed. You can also see whether cookies are attached to each request by examining the request headers. Try the same thing with other domains or port numbers (e.g., `http://bbs.demo.org:8888/origin.php`).
- `http://localhost/`: you can get different kinds of cookies here! After getting each cookie, check the response header and the cookies in the developer tools. Then go to `origin.php` and observe whether the cookies are sent with each request.

- `http://demo.org/login.php`: this is the login interface of DBS (I'm joking). Just put in your name and click on "Submit" and you will be gifted a cookie (you can see it in the developer tools).
- `http://demo.org/form.php`: after being authenticated as a DBS client at `login.php`, you can transfer money to your friend here. Put in the amount of money you want to transfer in the first field and the name of your friend in the second field, and "Submit".
- `http://localhost/csrf.html`: now there's an interesting website with a tempting "win big prizes" button. Just click on it to win big prizes. Please spend some time figuring out what big prizes you got and how you got them.
- `http://demo.org/form-safe.php`: this is almost the same as `form.php`. Please compare their source code.
- `http://localhost/csrf-safe.html`: here's another web page that wants to give you big prizes. See whether you can still get any here.

Under the Hood

The domains `localhost`, `demo.org`, `bbs.demo.org`, `blog.demo.org` all resolve to `127.0.0.1` so there is no difference in terms of the content served. The HTTP server listens on both ports 80 and 8888 but also serves the same things. What is important here is that your browser only cares about the *origin*, which includes both the domain and the port number. Since there are different combinations of domain names and port numbers, your browser will treat them as different websites. It does not matter to the browser whether they are served by the same machine with the same content. Hence with this kind of settings we can experiment with what the browser does when making different kinds of requests across websites.