Comp[redacted]:
Project 11: CSRF Prevention

[redacted]

Fall 2018

IMPORTANT: Do not share or discuss this document with anyone else in the course! Different students have different conditions. Please carefully read all instructions before you check out the code and begin working.

As a reminder, your grade here comes from participation rather than completion. We're trying to measure how well students do with different initial conditions. If you were to talk amongst yourselves or ask for help from TAs, then we'd be measuring something else entirely.

Do your best; do what you think is reasonable. Feel free to do web searches and follow good advice you might find on the Internet in addition to what we've taught you in class.

1 Deadline and Timing

This is a timed assignment. You will have three (3) hours to complete it once you check out the code and push your modified `README.md`. Please read over this full packet before you begin. (You'll find the clone link further down.)

Do what you need to do in advance to get yourself comfortable and undistracted, then go for it. If you hit the end of the three hours, please commit your work in progress and call it a day.

We ask that you use a stopwatch (perhaps an app on your phone). This way, if you get distracted by a phone call or whatever, you can pause the clock, then resume it when you're back.

Please complete your work no later than Sunday, November 4, 2018, at 11:59pm. After you complete the assignment, we have a brief survey that you should fill out (link below).

2 Fruit Market

This week, you will be working with the Fruit Market server. This is a single-user server that displays the current price of three different kinds of fruit and allows the user to buy

and sell fruit at the current prices. Buying and selling fruit changes the state of the user's current funds and fruit holdings, which are stored on the server.

2.1 Code

You'll find the server code in `src/main/java/edu/xxx/market/MarketServer.java`. The server state is defined by four static variables: `random` (a random number generator, used to generate random prices), `funds`, `prices`, and `holdings`. The `main` function launches the web browser, creates the handlers for server queries, and then enters an infinite loop to fluctuate the fruit prices once per second.

The Fruit Market server handles seven `GET` requests. `/` simply redirects to `/market/`, which serves the HTML for the site. `/funds/` returns the user's current funds, formatted as dollars as cents. The `/price/` and `/holdings/` requests each require an **index** `GET` query parameter, specifying the type of fruit (0, 1, or 2 for apple, orange, or banana, respectively), and return the price or current holdings for that fruit type. Note that so far, these requests don't modify any server state – they only request data from the server. Finally the `/buy/` and `/sell/` requests handle the buying and selling fruit, ensuring the transaction is legal and then updating the state accordingly. Like the price and holdings queries, these require an **index** query parameter to specify the fruit type.

The client-side JavaScript for the Fruit Market is located in `src/main/resources/WebPublic/market/market.js`. The script periodically updates the displayed values by issuing `GET` requests to the server and modifying the HTML document with the returned values. The six buy/sell buttons are wired to functions that tell the server to buy or sell.

2.2 CSRF Vulnerability

The Fruit Market server is currently vulnerable to cross-site request forgery (CSRF) attacks. For instance, an attacker could present a fraudulent web form (or even just an image embedded in another site!) capable of modifying the server's state. Your task is to modify the Fruit Market server and/or client to prevent CSRF attacks.

There are two files you should consider modifying: `MarketServer.java`, which contains the SparkJava server code, and `market.js`, which contains the client-side JavaScript.

Feel free to refer to the Wednesday lecture slides on security, which include a discussion of CSRF attacks. Similarly, feel free to use your favorite search engine to look up more on the topic. However, please do not ask another human for help! We want to measure how well you can solve the problem given the initial conditions that we created for you.

3 CSRF Protection Example

We have provided an example of CSRF prevention by giving you a "read-eval-print loop" (REPL) for the "Nashorn" JavaScript interpreter that's built into Java8. What's going on here: we're build a web server that accepts arbitrary JavaScript commands, and evaluates

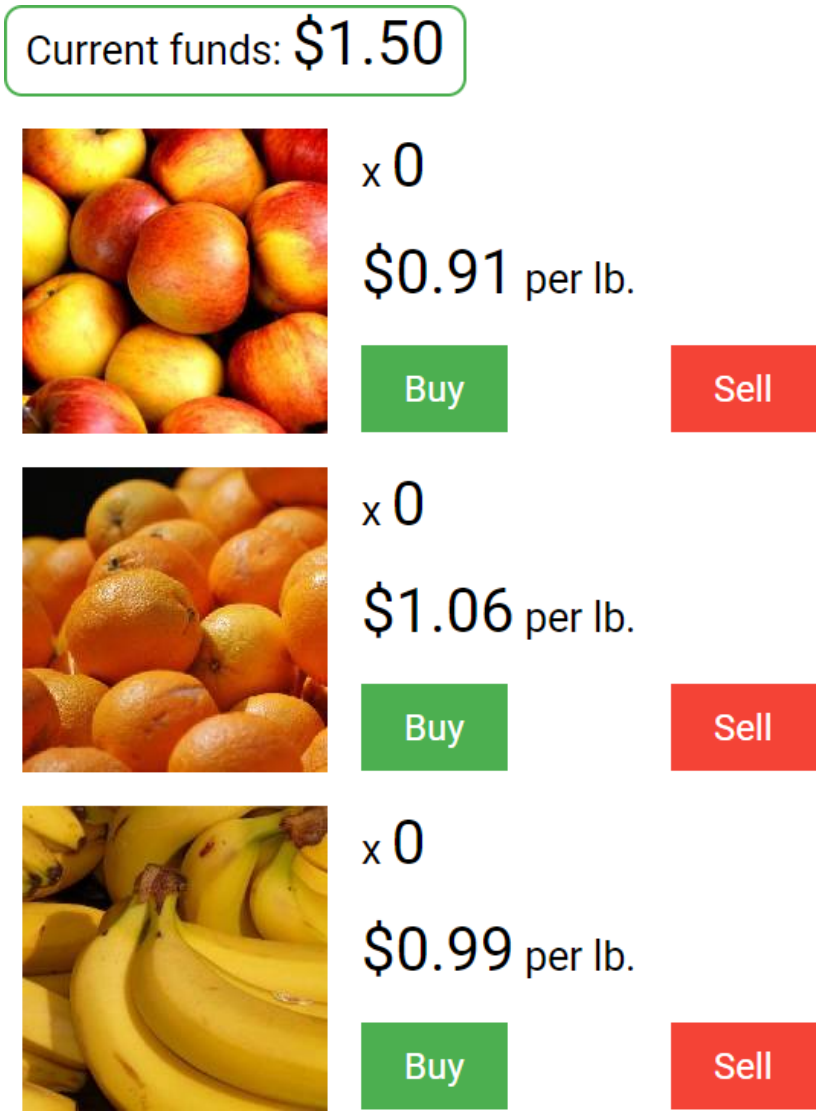


Figure 1: Fruit Market

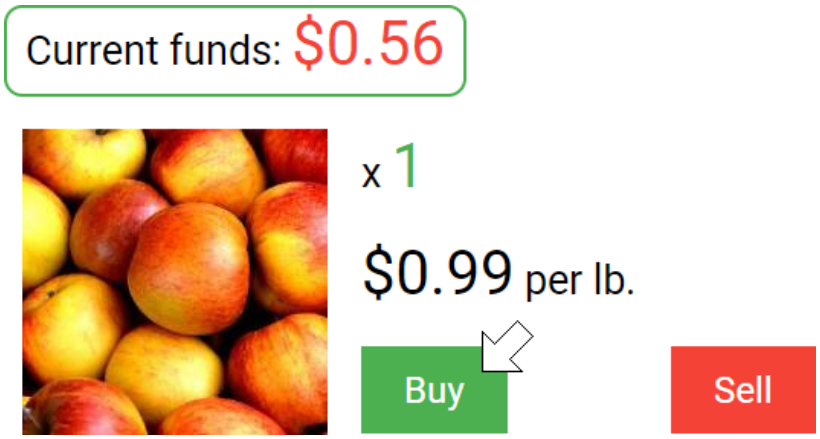


Figure 2: Buying fruit causes the counter for that fruit to increment and briefly turn green, and the current funds to decrease by the current price and turn red.

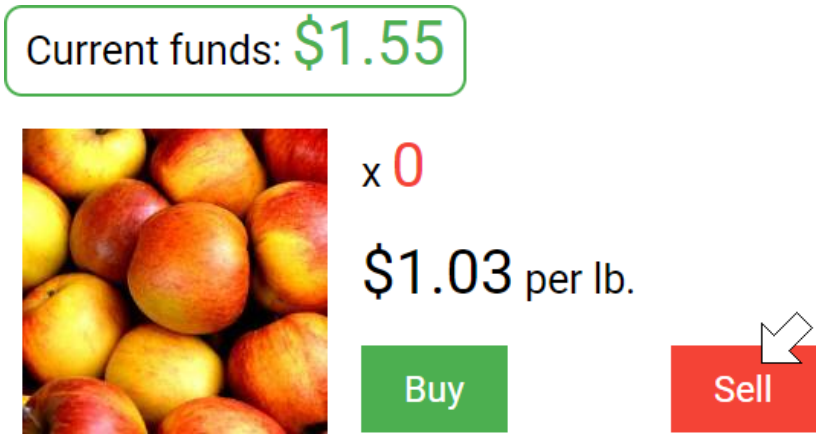


Figure 3: Selling fruit decrements the counter and turns it red and increases the current funds and turns it green.

Figure 4: Trying to buy fruit without enough money.

Figure 5: Trying to sell nonexistent fruit.

them on the server. If anybody on the Internet could connect to it or trick it into executing commands, that would be a security nightmare. Consequently, we've carefully engineered our JavaScript REPL to resist CSRF attacks.

The JavaScript REPL's web server is in `src/main/java/edu/xxx/web/JavaScriptRepl.java`, and its browser code is in `src/main/resources/WebPublic/jsrepl.js`. You'll notice how this code is quite similar to the RPN server and client code that you worked on for your project last week. You are encouraged to read through these files to get an idea of how to prevent CSRF attacks against the Fruit Market server.

4 CSRF Testing Tools

To help you test whether your solution successfully protects against CSRF, we have provided a copy of CSRF Testing Tools[1], a free tool for generating CSRF attacks to check servers for vulnerability. The files are located in `resources/WebPublic/market/CSRF-Testing-Tools`.

The `CSRF-Testing-Tools` directory contains a `README` with full instructions. When you open this file, you might notice that step 1a. says "Since I haven't set up hosting for the FormGrabber js file, you will need to host this file on a web site somewhere". We have already completed this step for you by modifying `formgrabber_bookmarklet.html` to point to a `formgrabber.js` hosted online, so you can ignore this step. You should be able to follow the rest of the steps as written to test your implementation.

We've confirmed that this tool works with all the latest versions of the Google Chrome browser.

[1] https://github.com/akrikos/CSRF-Testing-Tools

5 Summary of your work this week

5.1 What you'll do

- As soon as you check out the repository, edit the `README.md` with your name and NetID and commit and push your changes immediately. This gives us an independent measure of when you started working.
- Modify the server code (`MarketServer.java`) and/or client code (`market.js`) such that the Fruit Market server is safe from cross-site request forgery (CSRF) attacks.
- As soon as you finish the assignment, or three hours of work time have passed, commit and push whatever you have, even if you don't think you're done.
- Complete the post-study survey, linked below.

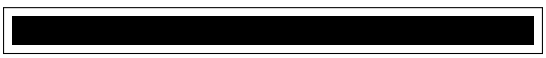
Mutation and use of `java.util` are allowed this week if you feel they're necessary. Obviously, we suggest you follow the same programming disciplines you've practiced all semester, but security is your top priority.

5.2 What you won't do

- Please don't discuss the assignment with other students or share any files outside of CompYYY.
- Your TAs have been instructed to ask you to "do your best" and otherwise let you work on your own. We want to measure how well you do.
- Don't lose track of your time. Use a stopwatch or timer app. Once you complete three hours of work, commit your work and call it a day.

5.3 Links

Now that you've read all the instructions and allocated a block of three hours, you may checkout your repository below and begin!



After you push your final commit, you will complete the following survey: