Challenge 3

Deadline: Check Piazza

Classic web application attacks remain a top concern, maybe *the* top concern, even though they've been extensively studied for decades. The causes and the cure are very well understood (Remember the golden rules? Input validation *CLAP* output sanitization *CLAP*.), but the attack surface is too large to manage effectively, and the impact on our lives is very real.

Here are your learning goals:

- Learn to recognize untrusted input.
- See the staple web application attacks SQLi, XSS, and CSRF in action.
- Gain hands-on experience penetration testing a proprietary web application blind.

Your Task

I present to you Dumpster®, the culmination of the advancements in e-retail and online community design, combining the best of Craigslist, eBay, and 4chan. What could possibly go wrong? Dumpster® is a free marketplace where anybody can trade anything! (and commit tax evasion)

Now that you have a few challenge wins under your belt, you are a fully qualified security expert, and you are tasked with performing a penetration test of this web application.

Here's what you need to accomplish:

1. Purchase an item **with an account you create.**
2. List a new item for sale **with an account you create.**
3. Delete one of the originally listed items.
4. There is a user **bob** who does a lot of talking, but no buying. *Make* **bob** buy something.

Simple, right? Not when you're working with an unprivileged account and no money.

That's all Folks!

Important Details

- Dumpster® uses **SQLite** for its database backend.
- **bob** reads his private messages frequently. He's very active on Dumpster®. But **bob** also deletes messages immediately after reading, so hold your multi-stage exploits!
- **bob** won't fall for social engineering. He's too smart.

- For your convenience, Dumpster cookies never expire. If you want to manually clean up all cookies for any reason, just send a GET request to `/reset` .
- You can run into unwinnable states in this challenge if you mess with the application too much and break everything. If you end up in that situation, send a GET request to `/ioncannon` . I will obliterate your database with a precision particle beam from orbit so that you can start with a clean slate. **This will reset all your progress.**
- Your tools may complain about a cookie named **session** expiring or being invalid. **Ignore that.** That's a quirk of the web application framework, and it has nothing to do with the challenge.

Accessing Dumpster®

You can find Dumpster® at 192.168.1.77, port **5001**. Dumpster® runs on the lab environment's private network. To access it from your personal machine, you need to set up a SOCKS proxy. Many SSH clients can do this easily. For OpenSSH: `ssh -4 -D <local_port> warhead`

...which connects to the *warhead* profile defined in your SSH configuration, and listens on *local_port* on your personal machine for connections. **The -4 flag is important; you need to set up the proxy for IPv4.** Now configure your web browser to use *local_port* as its SOCKS proxy. In Firefox: Settings -> General Settings -> Network Settings
You can achieve all that with any web browser, but I **strongly recommend** that you install and use Firefox for all web challenges. That's how they were play-tested.

The above configuration causes all your browser requests to be routed over warhead to their destination. That works for accessing Dumpster®, but what if you want to browse the Internet at the same time? There is no Internet access on warhead, so that won't work. You have three options:

- Use a different browser for Internet access.
- If Firefox is your daily browser, create a new Firefox profile to work on challenges and configure that for SOCKS. Keep using your everyday profile for Internet access as usual.
- Scrap the SOCKS idea, configure *port forwarding* instead. SSH clients can do this just as easily, but it's your job to figure out the right flags to use.

Tips and Hints

- There are no solution tokens to collect in this challenge. Just complete the above tasks, submit the challenge, and I will know.
- Before you start thinking about exploits, **identify all untrusted inputs** to the application. Then try to figure out which ones among them are unsafe, i.e., there isn't sufficient input validation or output sanitization. A vulnerability shouldn't be too far.
- If you ever get stuck, take a step back and repeat the above. Are there any other input entry points you might have missed? Inputs can come from anywhere you and your browser communicate with a web application. Use your programmer's instincts to

guesstimate how those inputs may get processed in the backend, how they may factor into authentication and authorization checks, how programmers can cut corners or make mistakes when implementing these checks…
- Don't limit yourself to the big 3 web attacks. In fact, one question here must be solved via pure exploration. Told ya *security by obscurity* was a bad idea…
- You are not completely in the dark. You can always create multiple accounts, try to exploit one another, and see the result from the other's perspective.
- Use a web request interceptor tool to see what's going on at the HTTP level. You can use something like OWASP ZAP or **Burp Suite** if that's your jam. But Firefox's built-in Web Developer Tools work just fine for our purposes here. **I still recommend that you learn Burp,** it is the standard tool in industry and an excellent addition to your résumé.

Submission

1. Create the challenge directory tree `~/submissions/challenge3/`
2. Create an empty `submission.txt` in the challenge directory. Or write a dad joke in it, up to you, but we need the file.
3. Run `submit challenge3`

Wait a few moments, and check the results. Don't forget to visit the Hall of Fame: https://www.khoury.northeastern.edu/home/kaan/rankings.html

Good luck, and happy pwning!