Raluca Ada Popa
Spring 2018

CS 161
Computer Security

Project 3

Due: April 20, 2018, 11:59PM

Version 0.5: April 3rd, 2018

Background

Your valiant efforts earlier this semester succeeded in stopping Lord Dirks from achieving world domination. Unfortunately he has achieved something way cooler: he founded a new hip Series-A funded startup known as "Snapitterbook". We don't know exactly what they do, but it sounds pretty cool – we hear it uses blockchain technology and artifical intelligence. But still, we must stop Lord Dirks before his IPO at the end of the semester!

Recommended Setup

A "superserver" is provided at `https://p3.kvakil.me/`. This will allow you to explore Snapitterbook without the need to download any software. However, Nick Weaver has managed to procure a local copy of the source code, available at `http://inst.eecs.berkeley.edu/~cs161/sp18/projects/3/project3.zip`. Nick Weaver recommends you download this source code anyway, since it might help you develop exploits.

NOTE: Please read the legal disclaimer at the bottom of the site. Your cooperation in using this shared resource is appreciated. If I notice abuse, I will be forced to take it down and we will all have to do the Alternate Setup which is not too hard but definitely way less cool.

Alternate Setup

Start by downloading the source code: `http://inst.eecs.berkeley.edu/~cs161/sp18/projects/3/project3.zip`. You will need the following software:

1. Python, version at least 3.3
2. Python `pip` for Python 3
3. Either the newest version of Firefox or Google Chrome

After you have installed the necessary software and extracted the source code, open a terminal and enter the Project 3 folder. If you are on Linux, macOS or Git Bash, run `begin.sh`.

If you are on Windows, run `begin.bat`. If everything runs successfully, you should see something like this:

```
* Serving Flask app "server"
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now open your web browser and navigate to http://127.0.0.1:5000/. If all goes well, you should see the home screen of "Snapitterbook". If you are unable to get this working, the instructional machines have all the necessary software.

NOTE: Only perform attacks on this local server. You violate campus policy and the law when directing attacks against parties who do not provide their informed consent!

Showing Your Exploits

As in Project 1, there are two parts to this project: the exploitation and the documentation. For the exploitation portion, you will submit HTTP archive (HAR) files to a Gradescope autograder. A HAR file contains a log of all the HTTP requests between you and the webserver. The autograder analyzes this log to check if you have successfully found an exploit.

Let's say that you have found an exploit. We will discuss now how to document it. Follow these steps:

1. *P3 Superserver*: Navigate back to https://p3.kvakil.me/ and request a new server to begin the exploit clean.
2. *Local Setup*: Restart the Flask server. Kill the old server by going to the terminal which has it and pressing Control-C. Rerun `begin.sh` or `begin.bat` as described above. Then reopen Snapitterbook in your browser.
3. *Firefox*: Right-click the webpage and click "Inspect Element". Click the "Network" tab. Check the "Persist Logs" and "Disable cache" checkboxes. Click the "Clear" button on the top-left (it looks like a garbage can).
4. *Chrome*: Right-click the webpage and click "Inspect". Click the "Network" tab. Check the "Persist Logs" and "Disable cache" checkboxes. Click the "Clear" button on the top-left (it looks like a circle with a backslash through it). Select "Use Small Request Rows" for "View".
5. Perform your exploit. This should include any setup that is needed for the exploit. If applicable, it should include a URL access which shows how the exploit could impact another user.
6. Return to the Developer Tools tab. Right-click any of the requests. Click "Save All As HAR" (Firefox) or "Save as HAR with Content" (Chrome).
7. To submit, ZIP all of your HAR files up and upload it to Gradescope under the "Project 3 Autograder" submission.

Your grade on the autograder completely determines your grade for this part; there are no hidden tests. Please note that this project is **entirely new**! If the autograder does not give you credit for an exploit which you believe is correct, please make a private post on Piazza. Include the URL of your Gradescope submission, and the HAR file which you believe that you should be getting credit for.

Help, Snapitterbook is Broken and It Can't Get Up

After performing some of these exploits, you might end up with a Snapitterbook which is beyond repair. If this happens, you just need to restart the server.

If you are using the P3 Superserver, simply go to https://p3.kvakil.me/ and request a new server.

If you are using the local setup, go back to the terminal where the Flask server is running, and press Control-C. Then, rerun `begin.sh` or `begin.bat` as appropriate depending on your operating system.

Getting In (5 points)

Unfortunately, nobody has been able to get a Snapitterbook account, it's *way* too exclusive! We're hoping to get access some other way. Can you figure out a valid username and password, so that we can do some other attacks? (*Hint*: look at the HTML source code for Snapitterbook. You should be able to do this by right-clicking and selecting "Inspect Element".) Submit a HAR file as detailed in the "Showing Your Exploits" section above. The HAR file should show you logging in with a valid username and password. This problem is marked on Gradescope as `test_zero`.

Getting Through (40 points)

Now that you have a foothold in the system, it's time to find vulnerabilities. There are **eight more** vulnerabilities contained on the Snapitterbook website. Find and exploit all eight. Use the instructions from "Showing Your Exploits" to create HAR files, and submit them to the autograder. Note that even though you have access to the source code locally, your exploits **should not** rely on this fact! In particular, **do not** assume that you know the contents of the database as initialized in `database.py`.

Of course, this portion of the project is extremely open-ended! Finding vulnerabilities can be very frustrating. Sometimes you think something should work and it does not. Sometimes you think something definitely should not work and it ends up working. It is all very romantic.

Musings aside, here are some tips on finding vulnerabilities.

- Review the relevant lecture slides. There is little hope in finding the vulnerabilities if you do not know they exist.
- By far the best resource is the OWASP Testing Guide. It is also extremely comprehensive and lengthy. To start your search, you should probably begin with the vulnerabilities we have discussed in class.
- You do not actually need to do anything special, just show the exploit is possible. For example, it is enough display a Javascript alert with a test message to prove XSS.
- Play around with the application a lot. Try invalid and malformed inputs. Have your pet cat run on the keyboard.
- You have access to a local copy of the application! Use that to your advantage. Look at error logs. Read the source code. Try to understand where possible vulnerabilities may occur, and begin your search there.
- We have not discussed every vulnerability in class. You will have to do some of your own outside research.

Moving On (40 points total)

The next part of this project is a post-mortem of your prior exploits, along with a further analysis. Answer the following questions. Submit your answers as a PDF to the Gradescope assignment "Project 3 Writeup".

Vulnerability Writeup (10 points)

Choose *two* of the vulnerabilities you found in the above section, which are of different types (e.g., do not choose two XSS attacks). Explain in detail how each vulnerability works, making specific reference to the server code. Explain what an attacker could do through exploiting each vulnerability. Explain in detail how to fix the vulnerability.

Weaponize Vulnerability (10 points)

Weaponize one of your exploits (or a combination of them). Explain in detail an exploit which allows the attacker to create a post which appears to be from the user `dirks`. Your weaponized exploit should work with minimal user interaction: do not assume that `dirks` will load any URL you give him. `dirks` will visit his own wall and profile, but he will not interact with them.

Other Issues (15 points)

In addition to the autograded vulnerabilities above, there are several other issues with Snapitterbook. It may help to take a look at the Python code itself to discover these issues (namely `server.py` and `auth_helper.py`). Explain each issue, and propose a fix for each. You need not give code for this part, a high-level overview is OK. For full credit, you will need to identify **three issues**. Submit **no more than five** issues.

Mandatory Feedback (5 points)

Your submission must include feedback about the project. What do you think could be improved about the project? What part of the project was your favorite?

Your comments will not in any way affect your grade, apart from needing to be written in English, be about the project, and consist of at least 4 valid, roughly syntactically and semantically sort of correct and appropriate English sentences, so you do need to provide at least some feedback to earn points!