CSE 484 / M 584: Lab #2

Web Attack Lab

Due: **Friday, May 22, 2020, 11:59pm**

Before you start

Please work in groups of up to 3 people (e.g., reusing your groups from Lab 1).
Please use the sign up form here: https://forms.gle/wfJfXLCuekq8wkYbA.
If you don't receive access within 24 hours, feel free to send us a friendly reminder.

Server Address

https://codered.cs.washington.edu/lab2

What To Turn In

In addition to successfully completing the exploits on our server, please submit a document (txt, word, or pdf) to Canvas that, for each exploit:

- Provides the payload you found (i.e., what you entered into the text box on our server)
- Provides a ~1 sentence intuition for how it works
- Any accompanying files you created (e.g., your PHP script)

Please note that there is an opportunity for partial credit here: if you can't get an exploit to work, please still submit what you tried and why you thought it might work.

Please join a Lab 2 group via Canvas before submitting.

Goal

The goal of this lab is to gain hands-on experience with penetration testing of web applications.

For this part of the lab, you are presented with three different scenarios. Each scenario asks you to perform a task you would not otherwise be able to complete as a regular, benign user. You'll have to figure out what vulnerability exists in each challenge, apply what you've learned in class, and craft a special *payload* to achieve your goal.

Scenarios 1-3 are listed below. They add up to a total of *27* points and *12* extra credit points.

(Optional) Back Story

Scenario #1: Pikachu, Meowth, and Cookies

Everyone likes cookies, and Pikachu and Meowth are no exception. As Team Rocket's 4294967296th evil plan, Meowth is going to purchase all the cookies within Pikachu's reach so Pikachu would eventually surrender and give himself in, but of course Team Rocket cannot win.

Having eavesdropped on their conversation, you learned that Team Rocket keeps the cookies they bought in 8 different safes and store the combinations to each of the safes in 8 different cookies Meowth carries with him. You also learned that Meowth set up a website to facilitate communications with his fans (if any). With these in mind, you want to find a way get Pikachu some cookies back before he faints from a lack of cookies... but how?

Scenario #2: Jailbreak

You have been put in jail due to a wrongful conviction. You have no one to depend on, and the only way you can eat that University Teriyaki again is to jailbreak. Physical locks are for the weak; as a former Jedi, you can easily break them with the Force. What bothers you are the digital locks that are connected to a central database. But then, some materials you've learned from CSE 484 flashes before you…

Scenario #3: Hack your 4.0

Having joined CSE 484, you realized a sad truth: there's no way you can get a 4.0 for the class. You've learned that the seemingly nice and friendly CSE 484 TA has no mercy and routinely fails students as a hobby, and that the only way to get a good grade is to surreptitiously hack into the gradebook and change your own grade.

However, your CSE 484 TA is like no other; there's no way his/her website can be vulnerable to any attacks, or so he/she says…

Points

The following is a breakdown of the points for each problem.

Scenario #1

Problem #

1. 2 points
2. 2 points
3. 3 points
4. 3 points
5. 2 points
6. 3 points
7. 4 points (**optional, extra credit**)
8. 3 points (**optional, extra credit**)

Scenario #2

Problem #

1. 2 points
2. 5 points
3. 5 points (**optional, extra credit**)

Scenario #3

Problem #

1. 5 points

Getting down to business

Now that you have read the motivational backstory, let's get started!

Helpful tools and setup

Browser

During the course of this lab, we recommend that you use Firefox. The server uses Firefox (IceWeasel) and your exploit might exhibit different behavior with another browser like Chrome (i.e., your code might work on Chrome but not on Firefox).

Also, disable extensions that may change how your browser handles cookies like ad blockers. If you use Firefox as your daily browser, and don't want to disable your extensions, you can install Firefox Developer Edition to have a separate, "clean" installation.

Note that protection tools that are built into the browser may interfere with this assignment. You might try turning off tools like Chrome's XSSAuditor and IE's XSS Filter.

Setting up your webpage

When doing XSS attacks, you will need to exfiltrate the cookie from the victim's browser to a location where you can retrieve the cookie. One easy way to do this is to set up a webpage that takes GET requests with parameters. The goal is to have a page such that when you navigate to `https://homes.cs.washington.edu/~<username>/cookieEater.php?cookie=secretCookieValue`, your page will record `secretCookieValue` so you can read it later. We will go through the steps to help you get set up.

1. Host your webpage at homes.cs.washington.edu, follow this link to read the FAQs.
2. Once you have figured out where to host your page, you will need to write some PHP (or any other server side programming language) that will retrieve GET variables. (Hint: This should not take more than 10 lines of code). Here are some hints on what you will need.
    a. PHP get variables
    b. PHP tutorial
    c. Using PHP to write to a file (useful for saving cookies)
3. Now, you can try out your cookie receiver by using your browser to navigate to `homes.cs.washington.edu/~<username>/cookieEater.php?cookie=secretCookieValue`, assuming your php file is named `cookieEater.php`. If your php script records the value `secretCookieValue` correctly, you can get started!

If your script does not work, you might want to check whether your PHP script can be run by the Apache server -- in particular, you might need to set the file permissions to world-readable. We recommend **chmod 644 for your php script** (world-readable but not world-writable) and **chmod 622 for the file to which you're writing cookies** (world-writable but not world-readable, so others cannot read your stolen cookies, but Apache can write to it).

Scenario #1: Pikachu, Meowth, and Cookies (XSS)

In this scenario, you will mount a cross-site scripting attack against all versions of the link sharing website, stealing the bot's (Meowth) login cookies, and using it to unlock the next level.

Helpful links for XSS:

- Javascript: http://www.w3schools.com/js/js_intro.asp
- XSS intro: https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29
- Tutorial on XSS: http://excess-xss.com/
- XSS Filter Evasion Cheat Sheet: link

As you progress through the problems, the filtering will get more challenging and you will have to think of more creative ways to evade the filters.

List of filters:

1. No filter
2. Filters 'script'
3. Filters 'script', 'style', 'on', and ' ' (space)
4. Filters 'document', '(', ')', '<', '>'
5. Filters '<', '>'
6. Filters s/[()<>+]//g (that's a regular expression that removes all the characters in the square brackets from the input string)
7. Filters s/[bcdfihzjrst<>]//ig (similar to above, but ignores case for letters as well)
8. Filters s/[0-9a-z]//gi (removes all numbers and all letters from input string)

XSS attack process:

- Login to the server, select a problem.
- Start by typing JavaScript into the "Send me an image link!" box
- See if you can get your browser to execute JavaScript (see if you can evade filters)
- Craft JavaScript to steal your own cookie and send it to a server (the one we set up previously)
- Enter your attack JavaScript into "Send me an image link!" box

- You will be redirected (the page will complain about the URL being invalid)
- Copy the URL at the address bar
- Go back one page
- Paste the URL into “Send me an image link!”
- Wait ~30 seconds for the bot to ‘visit’ your link
- If successful, your server will record the value of the bot’s cookies (authenticated=<something>)
- Copy the value, and use the Storage Inspector in Firefox’s developer tools to create the same cookie (with name=authenticated and value=<something>) for yourself
- Click “Open Safe #” on the top right corner, if you got the right cookie, the page will say “Congratulations! …”
- The button for the corresponding problem should turn green when you solve it
- Repeat for the rest…

Scenario #2: Jailbreak (SQL Injection)

For this scenario, you will need to perform a SQL Injection attack.

Note: Some attempted SQL injection attacks may be blocked by CSE’s web application firewall (WAF). If it takes a long time to load the page, you probably hit the WAF. You can try adding spaces or similar characters where possible, and when in serious doubt, check with the course staff if you’re on the right track. The correct solution can bypass the WAF.

Some helpful links:

- SQL (and SQL injection): http://www.w3schools.com/sql/default.asp
- Some more SQL injection: https://www.owasp.org/index.php/SQL_Injection

Scenario #3: Hack your 4.0 (CSRF)

For this scenario, you will need to perform a Cross-Site Request Forgery attack.

Some helpful links:

- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
- What is CSRF: http://talks.php.net/show/xss-csrf-apachecon2003/13

FAQs and Errata

- Your script **must** run on homes.cs.washington.edu (this is a feature, not a bug)! You'll need to use one of your group members' CSE accounts to host on homes.cs.washington.edu. More details for how to access and edit your files there can be found here: https://homes.cs.washington.edu/FAQ.html
- Q: Is the clicking bot working?
    - Yes (pretty sure), check using a previously working script for a different problem, email the course staff otherwise
- Q: My exploit is not working! (Possible answers)
    - Have you made sure that you can steal your own cookie successfully, by visiting the attack URL yourself in your own browser? If that does not work, then it won't work when the clicking bot clicks on it either.
    - Make sure your code is not creating a popup / new window
    - Make sure your attack is on `homes.cs.washington.edu`
    - (Especially for CSRF:) Make sure you are using https (not http) URLs for `codered` or `homes` in your exploit