Project: Flag

Due: Sunday, February 28, 2021 @ 11:59 pm ET

0 Introduction

As part of their training, the crewmates at the Crewmate Academy are required to take a course called `cs666`: "Secure Computer Systems". This professional development course at the Crewmate Academy has their own in-house course management web application, the *FLAG (Fast Lightweight Administrative Grades) Portal*. In this project, you'll exercise your web security knowledge and gain practice in *black box testing* to first discover the semantics of this unknown website, then figure out how to break it.

0.1 Requirements

CS166 students must complete the assignment described in Part I. CS162 students must complete the CS166 requirements as well as an additional component described in Part II. For CS162 students, Part I is worth 60% of the project credit, and Part II is worth 40% of the project credit.

0.2 Accessing the Website

Each student gets their own instance of the FLAG Portal to attack—you can visit your personal instance at `https://<your-login>.crewmate.academy`. The `LOGIN` and `PASSWORD` files in the `/course/cs1660/student/<your-login>/flag/credentials` directory contain the *access credentials* needed to view the site. (The `IP` file contains your website's unique IP address, but you should only use the IP if the domain isn't working.) This access credential scheme is outside the scope of the project's vulnerabilities. The application itself implements a distinct authentication scheme (described in Section 2.1) which *is* within scope.

0.3 Collaboration Policy Reminder

Large parts of this assignment can be completed easily once an idea is known. Please make sure that you're following the discussion guidelines for the projects as outlined in the course *Collaboration Policy* to avoid giving unfair help, or even damage, to other students' work.

Part I

The Portal

1 Assignment

You will discover and exploit **five *distinct* vulnerabilities** in the FLAG Portal. *Exploits* must allow you to perform a normally unauthorized action in the web application. For example, changing a grade or deleting a particular user's data would all count as successful exploits.

Having *distinct vulnerabilities* means all of your discovered vulnerabilities belong to different *vulnerability categories*. Appendix A contains a list of web vulnerability categories that we'll accept on this project. Refer to this list to make sure you've found distinct vulnerabilities—while it's okay to use a vulnerability to discover other vulnerabilities, you cannot receive credit for the same category more than once.

Vulnerabilities must also manifest within the website application to count. This means network vulnerabilities (such as sniffing unencrypted traffic or clogging the server to achieve denial of service) or human vulnerabilities (such as social engineering or phishing) are out of scope.

1.1 Vulnerability Reports

For each exploit, you should prepare a detailed report that covers the following:

- **Discovery (1 point)**: Identify the specific *vulnerability category* as well as state where this vulnerability appears in the website (on a particular page, across all input fields, etc.).
- **Impact (6 points)**: Explain how you exploited the vulnerability (how your attack works), and the impact of your exploit. Your explanation should justify why your exploit meets the vulnerability category's *criteria for demonstration* as outlined on the vulnerability category's page on the CS166 Flag Wiki (`http://wiki.crewmate.academy`).
- **Mitigation (3 points)**: Explain (from a technical perspective) how to repair the vulnerability without compromising intended site functionality and justify why this fix blocks your exploit (and exploits similar to it). Your fixes should go beyond the mitigations mentioned in the CS166 Flag Wiki—that is, you should aim to provide concrete advice as to how or where mitigations should be applied to the website's codebase. (While you don't have the website's code directly, you should still be able to come up with a mental model of how the website works and justify your mitigations using that model.) Some vulnerabilities may have no viable fix—but be sure to justify such an assertion.

You should also include any additional files needed to perform your exploit (code, payloads, etc.) in your final handin. Your report should allow us to *easily* recreate your attack from only your verbal (and written) explanations and submitted files.

1.2 Extra Credit

Each additional, *distinct* vulnerability you discover and exploit counts for extra credit—we will accept up to *three* additional extra credit vulnerabilities.

1.3 Grading

For this project, you will receive two scores: the *raw score*, the sum of points you earn from your vulnerability reports, and the *final project score*, a scaled value computed from your raw score. The *final project score* will be your final grade on the Flag project. The formula for how we will scale raw scores to final scores is in Appendix B.

2 Application Details

2.1 Accounts

Each CS166 student has their own FLAG account under their CS login with password `iam<username>`. For example, if your CS login is `jcarberr`, your FLAG username and password are `jcarberr` and `iamjcarberr` respectively. Feel free to log in as your fellow students or yourself.

Each member of the course staff also has a FLAG account under their CS login, but their passwords are not given. However, these passwords are poorly chosen—we certainly wouldn't use any of these passwords in real life, especially after taking CS166!

Finally, all grades and comments on the site are randomly generated, so don't feel bad if *your* account has bad grades on it—the website content reflects nothing about any particular student.

2.2 Assumptions

When writing exploits, you may assume users actively use the site. This means exploits that only work when a user logs in, submits a form, visits profile pages, or generally uses any of the features on the site are within the scope of the project. This also includes specially crafted links—you can assume you can get any student or staff member to click a link to any arbitrary site (for example, by emailing them). Make sure to document any user activity that your exploit relies on in your vulnerability report.

2.3 Resetting the Application

If you would like to refresh the website to its original state, you can do so by requesting the path `/reset.php`. This will delete, then recreate, the website. Finally, it will redirect your browser to `/setup`, which will generate the website's content and data.

If you request `/reset.php` and are not redirected to `/setup` (for example, if you are using a command-line tool such as `curl(1)`), you will need to request `/setup` before you can log in. (Also, if you think you've broken the website to the point where the `/reset.php` endpoint no longer works, email the HTA list and we'll reset the application for you.)

2.4 External Tools

You are allowed (and encouraged) to use the built-in "Developer Tools" features in Google Chrome and Firefox. Additionally, you may use Burp Suite to proxy traffic, inspect requests and responses, and modify and replay requests. Finally, you may need some way to receive HTTP requests—you can use services like Hookbin (`https://hookbin.com/`) or Postbin (`https://postb.in`) to create temporary endpoints.

Ask the HTAs before using any tools not listed above—using more fully fledged software or automated analysis tools goes against the spirit of the project and may result in a *Collaboration Policy* violation.

2.4.1 Modern Browers

Some modern browsers (like Google Chrome) attempt to block malicious requests or avoid running code that looks injected. Thus, a simple web attack may not work on your browser, but it could work on users using different browsers. For testing purposes, it may be useful to try to disable such protections or use a different browser (like Firefox) for a more reliable hacking experience.

2.4.2 Burp Suite

We've installed Burp Suite on the department machines—you can run it using the `cs166_burpsuite` command. (Alternatively, you can locally install the free Community Edition from `https://portswigger.net/burp/releases/professional-community-2020-12-1`.) Most relevant to this project is Burp Suite's "Burp Proxy" feature, which allows you to view, intercept, and modify HTTP requests and responses. While you're not required to use Burp Suite, you may find it useful on the project.

3 Deliverables

In the security world, attacks are only taken seriously when one can demonstrate that their attack actually allows one to perform unauthorized tasks in a clear and convincing manner. To give you the opportunity to exercise your security presentation skills (and give you a chance to practice the demo skills that the TAs exhibit in the lectures!), you must prepare a *recorded video demonstration* of all of your attacks for Flag as part of your final handin.

3.1 Video Demo

Your final handin must include an MP4 video file named `demo.mp4` in which you demonstrate each of your exploits against the FLAG Portal. The logistical requirements for this video are as follows:

- Your video must be *at most 12.5 minutes* in length. You should organize your demonstration in such a way that you can present *all* of your exploits within this time frame (including any extra credit exploits), as we will only grade exploits that are presented within the 12.5 minute window.
- We recommend that you use Zoom to locally record your presentation. Doing so allows you to easily record a screenshare of the FLAG Portal as well as the source code of any files or payloads that you need to demonstrate your exploits, and optionally also include a video of yourself presenting in the top-right (though this is not required). Zoom will also automatically export a video in the proper MP4 format. See `https://support.zoom.us/hc/en-us/articles/201362473` for instructions.
- You are free to edit your video in any way that you see fit, though you aren't required to. Similarly, you don't have to record your presentation in a single take, though you can if you want.

In your video, you should walk through all of the discussion points in each exploit's vulnerability report (*Discovery*, *Impact*, and *Mitigation*—see Section 1.1) and clearly justify why your vulnerability report satisfies all of the necessary conditions. In your presentation, you should assume that the grader of your video will not have read your `README.pdf` and is grading your project entirely on the contents of your video. In fact, we won't read your written reports in your `README.pdf` unless part of your presentation is confusing or unclear—though in that situation, we'll read the written reports to provide you with an opportunity to receive partial credit.

You should aim to convince your grader that each of your exploits would work against a *clean* instance of the FLAG Portal just from your presentation of that exploit. By "clean" instance, we mean an instance of the FLAG Portal that has been reset (see Section 2.3). This means that if your exploits may potentially interfere with each other, you should reset the application in between the presentation of your exploits.

3.2 Handing In

Your handin should consist of `demo.mp4` and a single PDF file `README.pdf` that contains written forms of your vulnerability reports *in the order you are presenting each vulnerability in your demo video.* You should also submit any code, files, or payloads needed to execute each of your exploits. Your additional files do not need to be named in any particular way as long as you make clear in your recorded demo and in your `README.pdf` which files are relevant to each vulnerability report.

As mentioned previously, your `README.pdf` is primarily for partial credit, and we won't read it unless part of your presentation is confusing or unclear. Thus, you don't *necessarily* have to write completely detailed reports in your `README.pdf`—however, we strongly recommend that you do since otherwise you won't be eligible to get partial credit if things go awry. Regardless of how detailed you decide to make your `README.pdf`, you must still include some kind of listing of the vulnerability categories you are presenting in your video in the order you are presenting them.

Once you're ready to submit, please upload your files to the appropriately named upload point on Gradescope—do *not* upload your files as a `.zip`.

Part II

Bob's Router

CS162 students must complete the following, additional problem.

In this problem, you will launch a multi-step attack against a network and explore how security holes can compromise other clients of the FLAG Portal.

4 Setup

Finding vulnerabilities in the FLAG Portal certainly was effective, but you're concerned about the traceability of your actions—surely someone could check the server's request logs and notice something was up. Performing a man-in-the-middle attack against the internet traffic of one of your TAs would certainly be a more covert way to break the the Crewmate Academy's systems.

Luckily, the FLAG Portal's sysadmin (Bob) uses an old home router which undoubtedly has vulnerabilities. (Part of the router's manual can be found in Appendix C.) If you could gain control of his router, you could intercept and spoof Bob's network traffic! Unfortunately, Bob's router doesn't accept incoming connections; it only allows local clients to make outgoing connections. In order to interact with the router, you'll somehow need to convince one of the computers on the local network to make requests for you.

You know that Bob frequently checks on the state of the FLAG Portal (and, as a result, refreshes the FLAG Portal's login page every 15 seconds)—maybe you can use this as a way to hack into his router?

4.1 Diagram

Figure 1 illustrates the various network connections and machines involved in this problem.

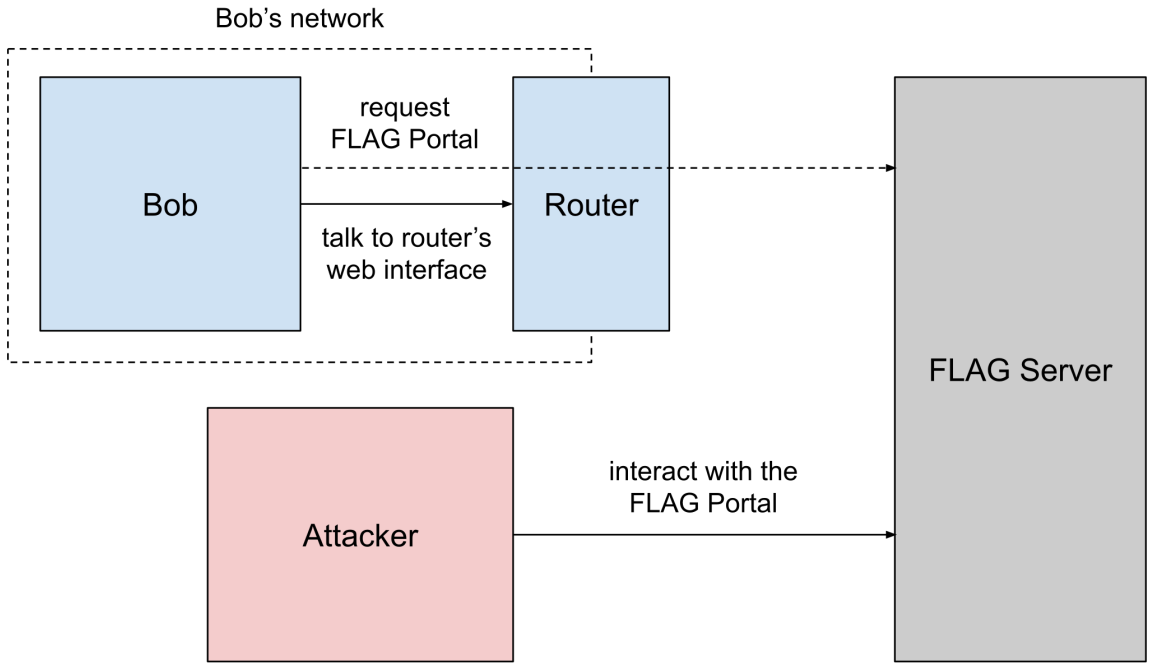

Figure 1: Network configuration for Bob's Router.

5 Assignment

5.1 The Attack

You will take advantage of Bob's visits to FLAG Portal's landing page to perform the following attack:

1. Execute arbitrary JavaScript in Bob's browser by using one of the exploits you crafted in Part I to inject JavaScript into the FLAG portal's landing page.
2. Launch a CSRF attack against Bob's router using your JavaScript exploit. Like many routers, Bob's router runs a web interface for monitoring and configuration purposes. Machines on Bob's local network can use the domain name `router.local` to access the router. (This domain will not mean anything to machines outside of Bob's local network.)
3. Discover as much as you can about the router and its attack surface. Using this information, figure out how to log into the router's web portal.
4. Once you've logged in, look for further vulnerabilities. Eventually, you should discover a remote code execution (RCE) exploit on the router itself (which runs Linux).
5. Finally, prove that you've successfully taken over the router by finding the *flag*. What constitutes the "flag" will make sense after you've poked around the router a bit.

One technical note—you might test your JavaScript injection attack by injecting an `alert` into the FLAG landing page. However, Bob doesn't understand that he needs to close alert popups before refreshing the page, so any `alert` *will* cause Bob to get stuck and stop refreshing the page. Because of this, you should avoid injecting `alert` functions or prepare to reset Bob (Section 5.3) if he ends up getting stuck.

5.2 Extra Credit

A *reverse shell* allows you to run arbitrary shell commands on a remote machine. For 7 points of extra credit towards your *final project score*, you may implement a reverse shell for Bob's router that satisfies the following requirements:

- Runs on any internet-connected machine (that is, it doesn't need to run on any particular computer).
- Allows the user to run arbitrary shell commands on the router.
- Streams, in real-time, the `stdin` of the locally running program to the `stdin` of the remote process running on the router as well as the `stdout` and `stderr` of the remote process back to the `stdout` and `stderr` of the local process. (In particular, it should not simply send the `stdout` and `stderr` after the process exits; otherwise, interactive programs won't work properly).

5.3 Resetting Bob

If you think Bob got stuck and stopped refreshing the FLAG Portal, please email the HTA list and we'll reset the system for you.

5.4 Handing In

Your handin will consist of two files: `FLAG`, which contains the flag you recovered from the router, and `README.pdf`, a PDF that contains a *detailed* account of the steps that you took in order to find the flag ("detailed" means you'll probably need to write a few pages). Additionally, you should submit any files that you needed to carry out your attack (code, payloads, etc.).

When you're ready to submit, upload your files to Gradescope. Do *not* upload them as a `.zip` file.

Part III

Appendix

A Web Vulnerability Categories

Below, we've listed every vulnerability category we could imagine coming up in a web security project like this. This means some categories may not necessarily have a corresponding vulnerability on the website.

While we've discussed some of these vulnerabilities in lecture, some are probably new to you (or might not appear in the same way you've seen before). Much of security involves learning about previously unknown systems, so we expect that you'll need to do your own research into some concepts covered in this project. If you find yourself at a point where you feel that you haven't been taught how to do something, that's okay! You should feel confident that you can do it if you set your mind to it.

We recommend the CS166 Flag Wiki (`http://wiki.crewmate.academy`) as a starting point for learning more about the vulnerability categories below—the Wiki also includes specific *Criteria for Demonstration* that your vulnerability demonstrations must satisfy in order to receive full credit. We also recommend using the *Open Web Application Security Project (OWASP)* at `https://www.owasp.org`.

- Bad Password Hashing
- Business Logic[1]
- Client-Hidden Sensitive Data
- Cookie Poisoning
- Cross-Site Data Access
- Cross-Site Request Forgery (CSRF)
- File Inclusion
- File Upload
- HTTP Parameter Pollution
- Insecure Direct Object Reference
- Path Sanitation Bypass
- Referrer-Based Access Control
- Reflected XSS
- SQL Injection
- Session ID Prediction
- Session Fixation
- Stored XSS
- UI Redress / Clickjacking

B Grade Scaling Formula

This scaling formula converts your *raw score* on the Flag project to your *final project score*. It guarantees that a raw score of 50 will result in a final project score of 100 for the Flag project. (For CS162 students, the "final project score" on Flag counts for 60% of the grade and the score on Bob's Router counts for the remaining 40% of the grade.)

$$\begin{aligned}\text{scaled score} = \min\left(-0.0152x^2 + 2.76x, 100\right) + \min\left(\frac{3 \cdot \max\left(x - 50, 0\right)}{10}, 3\right) + \min\left(\frac{2 \cdot \max\left(x - 60, 0\right)}{10}, 2\right) \\ + \min\left(\frac{\max\left(x - 70, 0\right)}{10}, 1\right)\end{aligned}$$

[1] Business logic vulnerabilities are considered on a case-by-case basis. If you find two or more business logic vulnerabilities, please consult the TAs to see if they count as distinct.

C Bob's Router: Manual

Below is the manual for the router that Bob uses to connect to the internet. Perhaps some (not all) of this information may be helpful for the assignment in Part II.

cisco Router Manual

C.1 Authentication

For convenience, your router comes preconfigured with one of the following default username and password combinations. Default credentials specific to your router are printed below the Ethernet ports.

- (`root`, `admin`)
- (`root`, `password`)
- (`root`, `secured`)
- (`root`, `123456`)
- (`cisco`, `admin`)
- (`cisco`, `password`)
- (`cisco`, `secured`)
- (`cisco`, `123456`)
- (`admin`, `admin`)
- (`admin`, `password`)
- (`admin`, `secured`)
- (`admin`, `123456`)

Remember to update the default password as soon as you receive the router. (For security purposes, the router's username is hardcoded and cannot be changed from the factory default username.) In order to update the password, you will need to manually reconfigure the router using the built-in keypad on the back of the router.

C.2 Information Overview

The Information Overview screen displays the current status of the various interfaces on the router, and the numbers of packets, bytes, or data errors that have travelled through the selected interface. Statistics shown on this screen are cumulative from the last 30 seconds that the router has been online. This page is accessible only from an authenticated administrator account.

Router status. If "OK", then the router is online and connected to the internet.

Packets dropped. The number of packets dropped (received by the interface and never forwarded) by the router.

Upload speed. The average rate that packets have been sent by the interface.

Download speed. The average rate that packets are received by the interface.

Default gateway. The system that the router interface must connect to in order to access the Internet or your organization's WAN.

Primary DNS. The default DNS lookup portal that the router interface must connect to in order to perform DNS queries.

The default gateway and primary DNS are read-only from the Overview page. In order to update this information, you will need to manually reconfigure the router using the built-in keypad on the back of the router.