Project: Flag

Due: Mar 13, 2025 @ 11:59 pm ET

Introduction

As part of their training, students at Blue University are required to take a course called `cs6660`: "Secure Computer Systems". This professional development course has its own in-house course management web application, the *FLAG (Fast Lightweight Administrative Grades) Portal*.

In this project, you will have an opportunity to exercise your web security knowledge by discovering the semantics of an unknown website and figuring out how to break it. You will play the role of tester who is investigating the website for vulnerabilities—for each vulnerability you find, you will write up a brief report on what you found, how you were able to exploit it, and comment on how the vulnerability could be fixed.

You are not required to develop fixes for the website—indeed, you even don't have access to the website's source code (unless you find an exploit to get it!)—instead, you can think of your report as something that could be presented to the FLAG portal's developers so they can address the issues.

Requirements

CS1660 students must complete the assignment described in Part I. CS1620 students must complete the CS1660 requirements as well as an additional component described in Part II. For CS1620/CS2660 students, Part I is worth 60% of the project credit, and Part II is worth 40% of the project credit.

Collaboration Policy Reminder

Please make sure that you're following the discussion guidelines for the projects as outlined in the course *Collaboration Policy*. Specifically, we encourage you to collaborate with others about how to work with tools, use the container environment, and discuss the high-level mechanics of various web attacks. However, the

process of discovering the vulnerabilities must be an individual effort. For more details, see the collaboration policy on the course website.

Part I

The Flag Portal

0 Assignment and Key Resources

You will discover and exploit **five *distinct* vulnerabilities** in the FLAG Portal. An *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 both count as successful exploits.

Wiki We've a wiki describing each type of vulnerability and some details and how each one works—we **highly** recommend reading this as a starting point to see what kinds of attacks you might be able to carry out. You can view the wiki here: `https://brown-csci1660.github.io/flag-wiki/`

Counting vulnerabilities Having *distinct vulnerabilities* means each of your discovered vulnerabilities belong to different *vulnerability categories*. Appendix A contains a list of web vulnerability categories that count as exploits for this project. You should refer to this list to make sure you've found distinct vulnerabilities. It's okay to use multiple vulnerabilities of the same type to help your discover other vulnerabilities on the site, you may only count one vulnerability of each type as one of your required vulnerabilities.

Scope Vulnerabilities must be part of the Flag portal website itself. This means network vulnerabilities (such as sniffing unencrypted traffic, overwhelming the server to achieve denial of service) or human vulnerabilities (such as social engineering or phishing to steal passwords) are out of scope.

Additionally, **attacks on the container infrastructure are also out of scope**, as these do not pertain to the flag portal website. That is, a vulnerability must not rely on using `docker` commands to "break in" to the container filesystem, as this is outside the "attack surface" of the website you are testing. While it's easy to break into the container this way, it is not in your best interest to do so—we are grading you on the web vulnerabilities you find and your demonstration of them, so attacks that are out of scope will not improve your grade.

0.1 Extra Credit

Each additional, *distinct* vulnerability you discover and exploit counts for extra credit—we will give points for up to *two* additional extra credit vulnerabilities, worth up to 5% of the total grade in total.

1 Background and Setup

1.1 Accessing the site

For this project, we have created a Docker image that you can use to run the FLAG Portal. We've prepared instructions on how to use it as a separate guide—for instructions on how to set up the container and use the portal, see here: `https://hackmd.io/@cs1660/flag-setup-guide`

1.2 Starter repository

You can create a repository and download the starter code using this link:
`https://classroom.github.com/a/zvpKmy99`

This repository initially just contains scripts to download and run the flag portal container environment. Since your primary goal is finding and writing about vulnerabilities, there is no stencil code for this project—instead, this repository is mainly a place to store and submit your `README` and any code you write.

1.3 Accounts

There are a number of student accounts that can be used to access the FLAG Portal. To start with, you can use the username `qmei` and password `iamqmei` to access the FLAG Portal **as a student**. Note that all of the student accounts have a similar username/password structure—each student account has a username with the corresponding password being `iam<username>`. Feel free to log in as `qmei` or any of the other students.

There are also a number of staff accounts on the FLAG Portal that have elevated privileges. We have not given you the passwords to these accounts. However, these passwords are poorly chosen—we certainly wouldn't use any of these passwords in real life, especially after taking this course!

1.4 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 a link). Make sure to document any user activity that your exploit relies on in your vulnerability report.

1.5 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.

Ask on Ed 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 not receive credit.

1.5.1 Burp Suite

While you're not required to use Burp Suite, you may find it useful for this project. For instructions on how to set up Burp Suite, see the Burp Suite lab: `https://hackmd.io/@cs1660/burp-suite-lab`

1.5.2 Which browser to use?

Some browsers like Google Chrome attempt to block malicious requests or avoid running code that looks injected, which can make the attacks we are performing more difficult. If you are concerned a browser may be blocking your request, you can check the browser's console (in Chrome, this is under View > Developer > Developer Tools)—if your attack was blocked, you will usually see a message relating to your attack that describes what was blocked.

For testing purposes, it may be useful to use Firefox instead, which is a bit more lenient on some attacks. If you want to use Firefox with Burp Suite, see `https://hackmd.io/@cs1660/burp-suite-lab#Using-Burp-with-Firefox-and-FoxyProxy`this section of the Burp Suite lab for details on how to set it up.

1.5.3 Resetting the Application

The Flag portal container is designed to be easy to reset to its original state so that you can easily repeat exploits, or revert the container if things go wrong. For instructions on how to reset your container, see this section of the Setup Guide.

2 What to submit

To present your vulnerabilities, you will submit a detailed readme with your *vulnerability reports*, which are a detailed analysis of each vulnerability you found and how it works. In addition, you will submit a short demo video demonstrating each vulnerability, so that we have a record. For more details, see the following sections.

2.1 Vulnerability reports

Your readme should be a single PDF file and should include a detailed discussion of the vulnerability you found and how it works. This is the primary way we will assess you work. There is no official length requirement, but please make sure you include enough detail to demonstrate your understanding (perhaps a few paragraphs, with diagrams or code snippets as necessary). In general, each vulnerability report should include the following components:

- **Discovery**: 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.).
- **Procedure**: Explain how your attack works—between your explanation and your video, you should provide enough detail to reproduce your attack and explain why it works. Feel free to include diagrams or images, or refer to your video.
- **Impact** Explain the overall consequences of the vulnerability—in other words, as an attacker, what does this let you do? Consider how your attack affects the users of the system. In addition, 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 Flag Wiki.
- **Mitigation**: 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 include more detail than what is mentioned in the wiki. For example, saying "sanitize inputs" is not sufficient—instead, what inputs should be sanitized, and how? If you believe a vulnerability has no viable fix, please explain why.

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.

2.2 Video Demo

To provide a video record of your attacks, you should submit an `.mp4` video file named `demo.mp4` that demonstrates each vulnerability. Some notes about this:

- Your video should be no more than 10 minutes and should just demonstrate each of your exploits– **you do not need to provide explanations in the video**. A simple screen capture while you perform each exploit is fine.
- The order of your vulnerabilities in your video **must** match the order they appear in the report. This helps us grade your work more easily!
- We recommend that you use Zoom to locally record your presentation. This 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 here for instructions.

- You are free to edit your video in any way that you see fit, though this isn't required. It's also not required to record your video in a single take.

Across your readme and video, you should aim to convince us that each of your exploits would work against a clean instance of the FLAG Portal just from your presentation of that exploit. This means that if your exploits may potentially interfere with each other, you should reset the application in between the presentation of your exploits.

2.3 Submitting your work

To submit your work, you should commit your readme with your vulnerability reports (as single PDF), your demo video, and any code, files, or payloads needed to carry out your exploits to your git repository and upload it to the "Flag" assignment on Gradescope.

If your video is too large to include in your git repository, please upload a shareable link to Google Drive and include a link in your readme.

Finally, please make sure that the order of your vulnerability reports in your readme are *in the order you are presenting each vulnerability in your demo video*–this helps us a lot when grading! Your additional payload/-exploit files do not need to be named in any particular way as long as you make clear in your readme where each file is needed.

Part II

Bob's Router (CS1620/CS2660 only)

CS1620/CS2660 students must complete the following additional problem, which involves a a multi-step attack against a network to explore how security holes can compromise other clients of the FLAG Portal.

3 Scenario

After reviewing your vulnerability reports on the FLAG portal website, the site's sysadmin, Bob, asks what else you can break. After a brief conversation, you've learned that Bob uses an old home router which undoubtedly has vulnerabilities, and think this would be a good target. (Part of the router's manual can be found in Appendix B.)

Home routers usually host a small website to configure them—older models are notorious for security problems. (We'll learn more about home routers later in the course.) However, you can't directly connect to the configuration website on Bob's router over the Internet—it only accepts connections from *inside* Bob's home network.

However, you've also learned that Bob has a computer at home that runs a script to check the state of the FLAG portal every 10 seconds (ie, to see if it goes down). Since you now know a *lot* of ways to compromise the FLAG portal, maybe you can exploit this as a way to attack Bob's router?

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

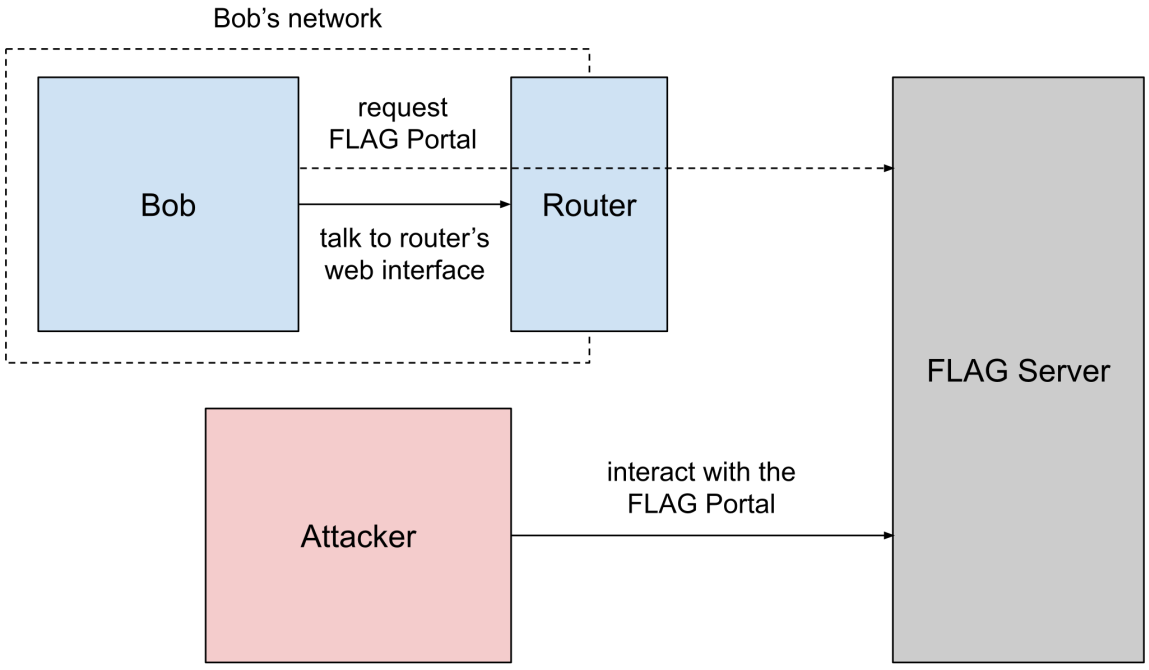

Figure 1: Network configuration for Bob's Router.

4 Setup

For a detailed guide on how to work with and set up Bob, see this section of the Setup guide. Then, return here for a description of how to think about the attack.

5 Assignment

5.1 The Attack

You will take advantage of the script that fetches the 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 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 that allows you to run arbitrary sever-side code.
5. Use this to obtain a *reverse shell* that allows you to run arbitrary Linux commands on the router. Once you can execute arbitrary shell commands, poke around and look for a file called `flag` to demonstrate you've taken control of the router.

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 if it gets stuck. For instructions on how to reset Bob, see the setup guide.

5.2 Handing In

Your handin will consist of two files: `FLAG`, which contains the flag you recovered from the router, and `readme-bob.pdf` that contains a detailed account of the steps that you took in order to find the flag (1–2 pages). Additionally, you should submit any files that you needed to carry out your attack (code, payloads, etc.).

When you're ready to submit, please commit your files to your repository and upload them to Gradescope.

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 CS1660 Flag Wiki (`https://brown-csci1660.github.io/flag-wiki`) 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

[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.

B 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

B.1 Authentication

For convenience, your router comes preconfigured with the following credentials:

```
Username: admin
Password: 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.

B.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.