Project: Flag

Due: Mar 10, 2023 @ 11:59 pm ET

0 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.

0.1 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 students, Part I is worth 60% of the project credit, and Part II is worth 40% of the project credit.

0.2 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

1 Assignment

You will discover and exploit **four *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://cs.brown.edu/courses/csci1660/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—while it's okay to use a vulnerability to discover other vulnerabilities, you cannot use a vulnerability in the same category more than once to count toward your required number of vulnerabilities.

Scope Vulnerabilities must also manifest within the website application itself. 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 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 `docker` command 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, 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 doing this will not improve your grade.

1.1 Vulnerability Reports

In a `README.pdf` file, for each exploit, you should prepare a detailed report that covers the following:

- **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.).
- **Impact**: 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 (`https://cs.brown.edu/courses/csci1660/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 go beyond the mitigations mentioned in the 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 give points for up to *two* additional extra credit vulnerabilities, worth up to 5% of the total grade each.

2 Background and Setup

2.1 Accessing the Website

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`

2.2 Starter repository

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

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.

2.3 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!

2.4 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.

2.5 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.

If you think you've broken the website to the point where the `/reset.php` endpoint no longer works, you can also restart the container from a fresh copy of the image. To do this, run:

```
./run-container --clean start
```

This will start a new container from the original image, erasing any changes made to the container filesystem.

Please let us know (through Ed, TA Hours, etc.) if neither of the above methods work for you when resetting the FLAG Portal.

2.6 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.

2.6.1 Modern Browsers

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.

If you are concerned a browser may be blocking your request, check the console in "Developer Tools"– usually a message will be printed indicating what was blocked.

2.6.2 Burp Suite

While you're not required to use Burp Suite, you may find it useful for this project. If you would like to use Burp Suite, you will need to install the free Community Edition on your local machine using this link: `https://portswigger.net/burp/releases/professional-community-2023-1-2`. Most relevant to this project is Burp Suite's "Burp Proxy" feature, which allows you to view, intercept, and modify HTTP requests and responses.

> **Note**: You are not required to install a CA certificate with Burp Suite in order to use it for this project. Installing a CA certificate is only required for intercepting HTTP traffic secured with TLS, which is not used in this project.

3 What to submit

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. For this project, you must prepare a *recorded video demonstration* of all of your attacks as part of your final handin, alongside a `README` containing your vulnerability reports.

3.1 Video Demo

Your 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 should be *at most 10 minutes* in length (shorter videos are perfectly acceptable!). In your video, you should only demonstrate each of your exploits–**you do not need to provide explanations**. Your `README` is the only place you need to include your vulnerability reports.
- 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.

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 2.5). 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 Submitting your work

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.

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 (CS1620 only)

CS1620 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.

Note: A separate guide about Bob's router will be released soon! For now, you can read about the mechanics of the attack and try it out if you want.

4 Setup

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.

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

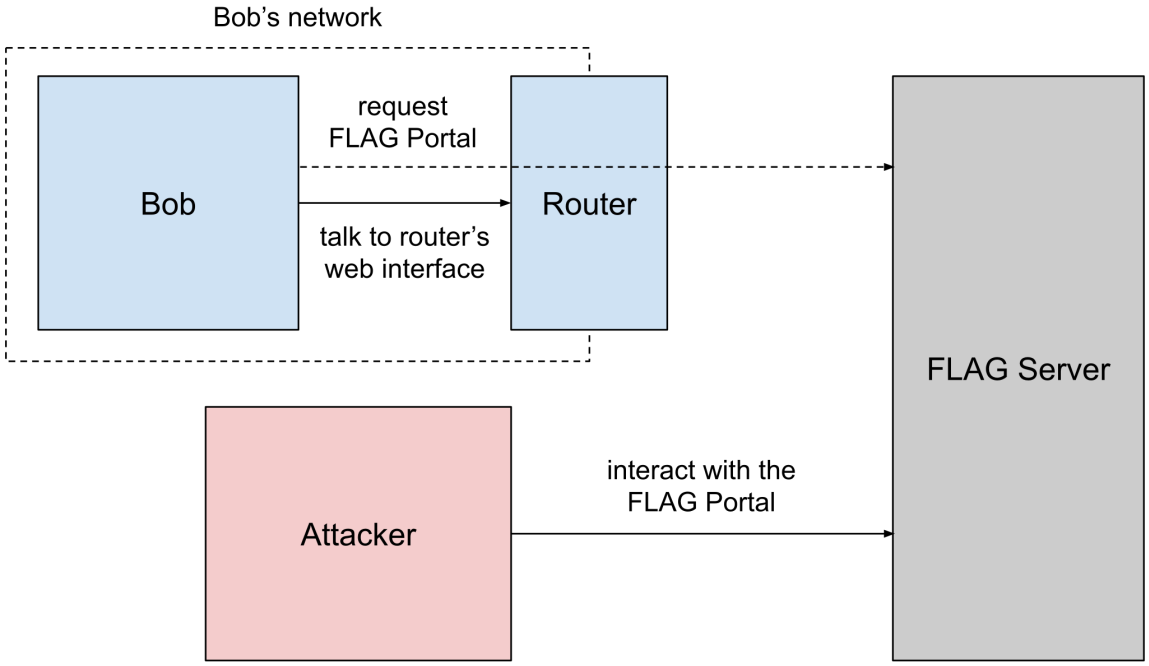

Figure 1: Network configuration for Bob's Router.

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 5.2) if he ends up getting stuck.

5.2 Resetting Bob

Similar to the FLAG portal, you can reset Bob's state by adding the `--clean` flag when starting the container for Bob's router, as follows:

```
./run-container --clean start
```

5.3 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.

6 FAQ

If you are performing the attack on a Windows machine, you might get the following alert from Microsoft defender:

you can prevent Microsoft from deleting your exploit files by clicking on the message, choosing the "allow on device" option, and clicking the "Start actions" button.

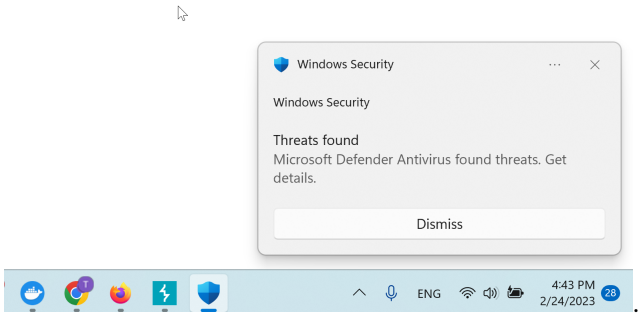

Figure 2: Microsoft defender alert

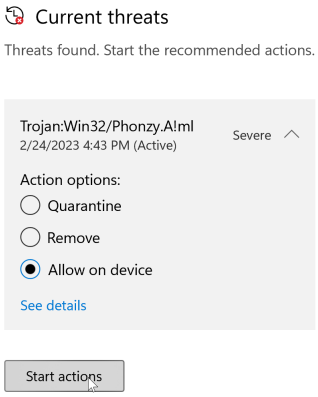

Figure 3: Allow file on device

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 (`https://cs.brown.edu/courses/csci1660/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.