Front-end Design for the Pwnable Claw Machine

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Computer Science,

By:

Charlotte Clark

Project Advisor:

Professor Robert Walls

Date: December 2022

Abstract

Capture-the-flag competitions are a popular educational tool for cybersecurity. We designed and implemented a web interface for a beginner-friendly capture-the-flag (CTF) to encourage participation in the cybersecurity program at Worcester Polytechnic Institute. The competition centers around a physical claw machine, where successfully solving challenges earns you a chance to use the claw and win a prize. We adapted an existing open-source CTF website to better fit our needs, incorporating new colors, theming, and functionality to create an exciting competition platform for students of all experience levels.

Acknowledgements

Thank you to my teammates Evelyn Dube, Avery Smith, Declan Murphy and Arthur Ames for their contributions to the Pwnable Claw Machine.

Contents

List of Figures

1 Introduction

Capture the flag competitions (CTFs) are a popular tool in the cybersecurity education community. Combining practical techniques with teamwork and competitive spirit, CTFs are a structured, safe way to gain experience hands-on security techniques in a controlled environment. CTF creators can design problems to guide students through different techniques in a way that is engaging, without breaking the law.

WPI has a small but strong cybersecurity community. Courses in cybersecurity are primarily targeted to seniors due to the amount of background knowledge needed to succeed. Because of this, many underclassmen do not consider cybersecurity as a potential area of research, unless they arrived at WPI with a preexisting interest in systems programming or other security topics. Our goal is to make cybersecurity more appealing and accessible to students without prior experience in the subject. By exposing students to cybersecurity earlier in the academic career, they may find themselves interested in taking advanced coursework in the subject as upperclassmen. To achieve this, we are working to create an eyecatching, beginner-friendly CTF competition to increase awareness of the cybersecurity programs at WPI. We want to curate an opportunity to explore cybersecurity with a low barrier to entry, and hope to spark genuine interest from students who may be intimidated by the reputation of systems programming and cybersecurity.

Our design centers around a physical claw machine to be displayed in Fuller Laboratories. However, rather than quarters, this claw machine only accepts payment in the form of CTF flags. Students visit the associated website, solve challenges, and earn points which can be exchanged for attempts at the claw machine. In this report, we discuss the development and theming of the CTF website, from both technical and aesthetic perspectives. This report is part of a larger effort to construct the claw machine and other aspects of the site, to be submitted at a later date.

2 Capture the Flag Structure

Cybersecurity Capture the Flag competitions typically fall into two categories: jeopardy-style and attack defense [11]. However, both types of CTFs usually feature teams competing for points, where the team with the most points at the conclusion of the event wins.

Jeopardy style CTFs are centered around independent challenges. The puzzles are created ahead of time, and may have different difficulty levels and point values. In this scenario, teams earn points by defeating the challenges created by the CTF developers. It is purely a race to earn as many points as possible. This is in contrast to attack-defense CTFs, where teams compete head-to-head. One team has to defend a resource, while the other is trying to gain access. Defending points are earned by successfully patching vulnerabilities, and attacking points are earned by successfully retrieving data.

For our CTF, we will be adopting the jeopardy style of competition. Because our participants will have highly variable skill levels, having teams compete directly would not be an enjoyable experience. Additionally, our CTF is open ended rather than being bound to a fixed time period. Since WPI students are often busy with coursework, having challenges available around the clock will encourage students to participate on their own time, as blocking off an entire weekend for a competition is often unfeasible. Completing challenges independently enables participants to compete asynchronously, increasing the accessibility of our CTF.

3 Background

PicoCTF is a free cyber-security education program run at Carnegie Mellon University. It utilizes the jeopardy-style CTF format to expose students to challenges curated by security and privacy experts [16]. In addition to being free to participate, in 2018 they released the

source code of their web-based competition software on GitHub [17]. While the existing implementation supports many features, such as inter-institutional competition, teams, and classrooms, for usage at WPI we wanted to both simplify and extend functionality of the PicoCTF framework.

3.1 Existing Infrastructure

Before discussing the modifications we made to the codebase, first we need to explore the design of the original site. PicoCTF utilizes several different technologies throughout the web interface. At the highest level, Jekyll is used to compile a collection of scripts and HTML files into layouts specified by templates. The majority of these scripts are generated by compiling React into vanilla JavaScript, which contains both logical elements and the UI code. These React files utilize the ReactBootstrap library for easy UI components with consistent styling. This combination of technologies has both benefits and unique quirks that we adapted to throughout the development process.

Jekyll

Jekyll is a Ruby Gem that is used for the development of static sites. Its templating system makes it ideal to use for sites like blogs, where many pages share a common format. The key components of a Jekyll site are a _layouts folder containing templates, HTML files for page content, and front matter at the start of each HTML file which specifies the page title, which template to use, and other variables.

Example of Jekyll HTML Syntax

```
---
layout: default
title: Homepage
foo: bar
scripts:
    - /js/script.js
---
<div class="page-content">
    <h1> Hello, world! </h1>
    <p>
        This is an example of an HTML file using Jekyll.
    </p>
</div>
```

The use of layouts allows developers to avoid rewriting boilerplate sections which are needed across several pages of their site. In PicoCTF, the main use of the templating system is to share the same metadata and navigation bar across the entire site. The metadata attached to each page includes the language, character encoding, icons, and scripts. While many of the scripts are shared across all pages, the layout also allows for pages to include additional scripts, such as files containing the React components used within the page. These scripts are included in the front matter of the specific HTML file.

React

React is a JavaScript library for building UI components. Developers can create customized, modular UI elements which can then be reused throughout the site. According to Stack-

Overflow's annual developer survey, React is the most popular front-end web library [14], far ahead of comparable technologies such as Vue and Angular.

React's component-based system is useful when designing interactive applications. It uses a unique JSX syntax, which extends existing JavaScript syntax to include HTML-like code [12]. This makes it easier to associate logic with the rendered output. These JSX files are then converted into vanilla JavaScript, which can be run in the client's browser.

PicoCTF uses React for all UI elements throughout the site. However, it still uses Jekyll as the foundation. This is unusual, because typically React is used for the entire site [12]. PicoCTF integrates the React components with Jekyll templating by converting the JSX into JavaScript first, and then including the resulting scripts in the Jekyll front matter as needed. Detailed steps for compiling and running the front end can be found in Appendix A.

Other Dependencies

Other libraries are also used throughout the code, but are less critical to the overall structure of the site. For instance, Bootstrap components are used for most of the UI, and jQuery is used to map React components with where they are placed in the HTML. However, it is important to note how libraries must be included in the site.

Because we are dealing purely with the front end, all of our scripts must run directly in the browser. This means that we cannot require third-party packages. However, packages are essential to developing for the web. To work around this issue, we download the JavaScript source code for the libraries we need, and add it into the project in the `/picoCTF-web/web/js/libs` directory. Then, we can include those files as scripts in the site's HTML. In our context, this means incorporating it into the Jekyll templates or front matter.

4 Design and Implementation

Our Goals

The primary objective is to integrate a physical machine with the PicoCTF software. We needed to connect, manage, and serve custom pages to the Claw Machine. In this report, we focused on the front-end development goals. Specifically, this included:

1. Adapting PicoCTF functionality to suit our needs
2. Creating a consistent user experience across both the claw machine and the accompanying CTF website
3. Streamlining the account creation process and integrating with WPI's Azure AD SSO

4.1 Expanding upon PicoCTF

Our greatest distinction from the existing PicoCTF competition is the inclusion of the claw machine. This means that there are two ways competitors interact with our CTF: online, and at the claw. In order to unify these experiences, the website will also serve as part of the user interface for the physical machine. While users will access the site to submit challenges from their personal devices, scoreboards and other information will be displayed on a screen inside of the claw machine.

The existing scoreboards page of the website would not be suitable for the claw machine display. Like the rest of the site, it utilizes a horizontal layout, navigation bar, and relatively small text. To rectify this, we created a new scoreboard page that does not include the navigation bar, optimizes the layout for a vertical monitor, and uses large text and bright colors. This scoreboard lists the users in order of the number of points earned, displaying

their username and point total in an arcade-game style layout. We used the same fonts and colors on this scoreboard as well as the main site, which further serves to unify the experience.

4.2 User Experience Design

Because this site is meant to accompany a physical machine, we needed to establish a clear vision for a shared aesthetic. This is important to ensure that the site and machine are cohesive and easily identifiable. Early in our discussions of the project, we decided to adopt an "90's roller rink/arcade" design goal.

Figure 1: Moodboard created to refine the desired design aesthetic

We chose this style for several reasons. Primarily, we wanted a fun and eye catching design that would easily catch the attention of busy students on their way to class. However, we did not want the design to be overwhelming. The arcade color palette is bold yet familiar, making it a strong choice to satisfy those objectives.

This aesthetic is characterized by deep blue blacks, bright neons, and lots of glow. We wanted to capture this energy while still being accessible and easy to use. To achieve this, we decided that it would be best to preserve the overall structure and layout of the website, which is already easy to navigate. Instead of overhauling all of the React components,

we simply added extra CSS to reach our aesthetic goals. The changes made were largely superficial, such as color and font choices. This means that the shape, spacing, and layout of the Bootstrap components remain the same. For color changes, we followed Material Design dark theme guidance [9] to ensure that there is sufficient contrast between background and text.

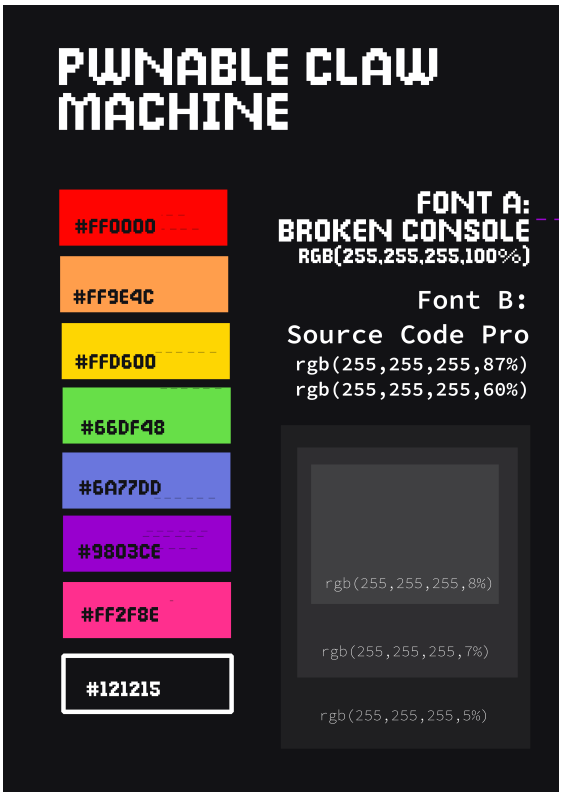
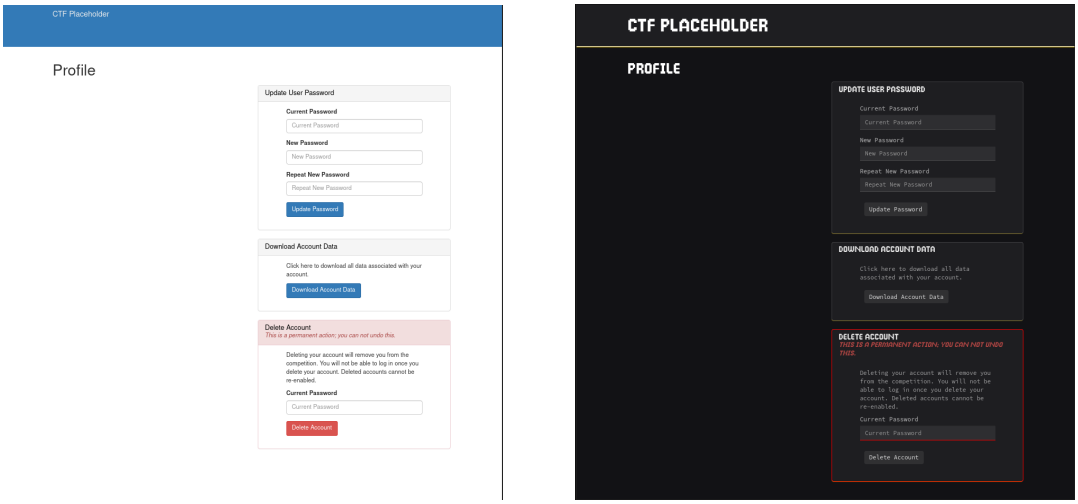


Figure 2: Guide to the colors and fonts used in the redesigned site

However, we did deviate from both Bootstrap and Material Design aesthetics when it came to fonts. We selected an 8-bit style font called Broken Console to be used for headings throughout the site. Its playful and retro appearance fits with our claw machine design goals, and helps unify the website with the machine. For non-heading text, we chose the monospace font Source Code Pro, which complements Broken Console nicely and is easier to read at smaller font sizes.



(a) Original design (b) Updated design

Figure 3: Before and after site redesign

4.2.1 Story

Additionally, the team decided that our CTF challenges would be part of a greater storyline. This means that we need to tie in references to the plot as well as callbacks to the arcade machine themeing. Our story lead determined that the overarching storyline would involve ambivalent, but incompetent aliens from outer space. In the aliens' research of human culture, they determined that we worship internet cats as deities, and therefore take on a feline form when communicating with humans.

To incorporate those ideas, we created animated pixel-art sprites of cat characters that appear sparingly throughout the site. Each cat character is associated with a specific type of challenge, and appears on information and warnings related to problems in that category. While originally we wanted to use easily recognizable internet cats, we found that copyright could be a potential issue [13][15]. For this reason, we decided to develop original cat images for use on our site.

We created four cat images, each with a unique personality and usage. Our first cat is a timid black cat that appears when a problem is about to expire. While black cats

Figure 4: Four cute cats

are typically thought to be bad luck, we hope his warnings will come in handy.

Other cats, such as our elegant seal point Himalayan, appear when users view the full description of a problem. These descriptions are phrased as if they were directives coming from the cats themselves, and serve to further reinforce the connection between the storyline and the CTF challenges.

These images were created using Piskel [3], a web-based pixel art tool. We chose to use Piskel because it is free and open source, and supports all the features we needed to create animated sprites of the cats. Piskel can be easily accessed from your web browser, and creating and exporting works is simple and free. Other popular tools, such as Aseprite [2], must be installed locally and cost money unless we were to build it from source. These two factors alone made Piskel a clear choice for the development of our sprites

4.3 Azure Integration

Aside from modifying the appearance of the site, we were also responsible for updating the login and account creation process for the CTF. PicoCTF is designed for competitors around the world, from different institutions. Their current registration process requires a large amount of information from participants, such as location, school, gender, and age. However, for our purposes, we can expect that practically all CTF participants will be WPI students. We wanted to integrate the login and account creation process with WPI existing Azure Single Sign-On (SSO).

By using WPI's SSO system, we lower the barrier to entry of the competition. It makes signing up to participate in the CTF as easy as checking your email, instead of filling out a somewhat-lengthy form. It also helps to associate user accounts with the students that they belong to, so that we can more effectively gather demographic information in the future. Additionally, it ensures that students can only create one account to play. This is important because it reduces the risk of students creating multiple accounts to solve easy challenges repeatedly.

While we expect that that majority of students will register with SSO, we also wanted to allow individuals without WPI emails to compete. We expect that this functionality will mostly be utilized by high school students visiting WPI, or students from other universities. For this reason, when a user visits the site, they are prompted to choose between signing in with WPI SSO or going through the standard registration process.

In order to enable Azure SSO for the CTF, we had to register our app with Azure. For more information about registering applications for SSO, see Appendix B

Then, we added support for SSO into the CTF's frontend. Azure SSO uses the OpenID Connect (OIDC) protocol, which is an extension of OAuth 2.0 [8]. OIDC works similarly to OAuth in most ways, but it includes more data about the identity of the end

user [6]. The key addition is the ID Token, which contains profile information about the end user such as their name and email address. This is in addition to the OAuth's Access Token, which grants users to access a specific resource [7].

To support this protocol, we decided to use the oidc-client-js library [10]. While it is currently no longer being developed, it is the most compatible with the existing PicoCTF codebase. Other libraries, such as Microsoft's MSAL.js [1] would be more difficult to build and save as part of our project. Since we cannot use any import statements, it was very important that a built, minified version of the package be readily available. While CDN-hosted versions of MSAL are available, that method of inclusion is inconsistent with other packages used throughout the site.

It is important to recognize that this implementation of SSO uses implicit flow to retrieve user data from Azure. Implicit flow is largely not recommended for many use cases due to security vulnerabilities. The data is returned via HTTPS without confirmation that it has been received by the correct client, it is vulnerable to man in the middle attacks among others. However, it is not an issue for our use. Because we are using OIDC only to retrieve the ID Token, and not an Access Token, the risks are no longer significant [4][5]. However, it is a potential area of future improvement to convert the process to use Authorization Code Flow with PCKE instead of implicit flow.

In addition to adding front end support, we also created additional endpoints in the backend to register users who sign in with SSO. These methods are essentially small tweaks on the existing methods to support logging in, but with changes to the parameters. PicoCTF's original login API uses username to identify users, but for our case, we want to identify users by the email address retrieved from Azure. Since these users also do not have a traditional password, we updated the MongoDB schemas to list password as an optional parameter. However, we were careful not to remove any functionality so that the original PicoCTF endpoints will continue to work as expected, for users without WPI email addresses.

5 Conclusion

Over the course of the fall 2022 semester, our team successfully developed a unified design objective for the Pwnable Claw Machine, redesigned the existing PicoCTF web interface, and began the process of integrating with Azure AD for SSO. Our design is an amalgamation of 90's arcades and internet cats, designed to capture the attention of busy students going about their day-to-day activities. Our redesigned site incorporates these ideas in a moderated fashion to maintain accessibility while still being closely coupled with the claw machine.

The Azure AD integration is beneficial in two ways. It allows students to quickly and easily sign up for the competition, while also making it easier for CTF administrators to moderate users. By tying accounts to their WPI identities, we mitigate the risk of students abusing the system. Both of these factors will contribute to a more enjoyable experience for participants.

Working on this project has proved to be a surprisingly valuable experience. Even with prior experience in web development and design, modifying PicoCTF to suit our needs was challenging and engaging. While coursework can expose you to the development process at the basic level, there is little that can prepare you for working on a large, year-old codebase originally written by strangers. Working our way through the idiosyncrasies of the PicoCTF source code was a greater challenge than the changes we needed to make to it. This experience has deepened our understanding of software engineering and the importance of documentation.

We hope that the Pwnable Claw Machine will be a delightful way for students to engage with cybersecurity at WPI. With that being said, this report represents only of a small fraction of the efforts needed to bring it to production. Work will be continuing through the rest of the 2022-2023 school year to put finishing touches on the web front end, refine the storyline, develop challenges, and construct the physical machine.

Appendices

A Running the Frontend

Since I am unable to run the full site on my computer, I figured out how to run only the frontend of the site alone. This way, I am able to better integrate my own development with the existing frontend architecture.

Installing Ruby and Jekyll

Jekyll is a tool used to reduce the amount of boilerplate HTML code needed for each page. Unfortunately, Jekyll is a Ruby Gem.

Ruby can be installed with `sudo apt install ruby-full npm`

Once Ruby is installed, navigate to the `PicoCTF/picoCTF-web/web directory`.

In this directory, run the command `sudo gem install jekyll bundler` to install Jekyll

Use the command `bundle init` to create a Gemfile in the web directory. Gemfiles are the Ruby equivalent of something like a `package.json` file.

Edit the newly-created Gemfile to include the line `gem "jekyll"`, which tells Ruby that Jekyll is needed for this project.

Then, you can run the command `bundle` to install Jekyll and any other Ruby dependencies for your project.

Installing Node and Babel

Babel is a package that can be used to transpile JSX files into vanilla Javascript. There are several React components used in this project, and they need to be transpiled before we can

actually use them.

First, install Nodejs and NPM with `sudo apt install nodejs npm`

Then, you can once again navigate to the `PicoCTF/picoCTF-web/web` directory.

The original authors have provided their `package.json` file, which includes the Babel dependency. To install Babel and any other Node dependencies, run the `npm install` command.

Transpiling the React Components

While in the `PicoCTF/picoCTF-web/web` directory, run the command `npm run jsx`

This is a script that the original authors specified in the `package.json` file. It uses Babel to transpile the React `.jsx` files in `PicoCTF/picoCTF-web/web/jsx` into usable Javascript. The resulting Javascript files can be found in `PicoCTF/picoCTF-web/web/js`

Building the Site with Jekyll

In the `PicoCTF/picoCTF-web/web` directory, make sure that you have your Gemfile containing the Jekyll dependency, and that Jekyll is installed.

Run the command `sudo bundle exec jekyll build` to build the pages of the site.

The fully built HTML files can be found in the `/srv/http/ctf` directory of your computer. This can be found in the root directory of your file system. However, CSS and other styles do not appear when opening these HTML files directly in your browser.

To view the pages of the site complete with styling, we have to backtrack. Going back to the `PicoCTF/picoCTF-web/web` directory, you can run `sudo bundle exec jekyll serve` to run a development version of the site at `localhost:4000`

B Registering Applications in Azure

In order to use WPI's single sign-on for our CTF, we have to register the website in Azure Active Directory. When creating a new registration, there are specific configuration options needed in both Azure and the application code to establish single sign-on functionality

Single Sign-on

Single sign-on (SSO) is a system that allows for streamlined signup and sign-in processes for supported applications. Instead of having to create several different sets of usernames and passwords for work-related resources, an identity management platform can serve as an intermediary. Rather than signing in to each app separately, users sign into the identity management platform once. Then, when they attempt to access resources, the application checks with the identity management platform to verify that the user is signed in. The user is then either redirected to sign into the identity platform, or to the requested resource.

However, both the application and the identity platform must be configured to recognize each other. The identity platform must know what information the application will request about users, and the application must know what identity platform to send the request to. This process varies from platform to platform, so we will only be focusing on the platform relevant to WPI.

Azure Active Directory

Azure Active Directory (AD) the identity and access management platform used by WPI. All users with WPI-issued email addresses have permissions specified in AD for access to certain resources and applications, such as Workday or Canvas. When you attempt to view these sites, you simply sign in to Microsoft with your WPI credentials to gain access. These

applications have been registered with AD. Our goal is to register CTF website with AD as well, so that students can quickly and easily access the competition. This is a multi-step process that requires both configuration in AD and modifying the code of our application. Luckily, each step on its own is relatively straightforward.

Creating an Application Registration

From your web browser, navigate to `portal.azure.com`. If you are not already signed into your WPI account, you will be prompted to do so.

Then, under Azure services, select Azure Active Directory. Your page should now show an overview of WPI's Active Directory.

In the left hand sidebar, under Manage, select Application registrations. By default, this page will show you a list of applications you have registered before. If you have not registered any applications, this will be empty.

In the top left corner, select the New registration button. You will be taken to a form for the initial setup of the application registration. For now, you will choose a name for the application, who will have access to the application, and the redirect url for your website. The redirect url is the page of your application that users will be taken to after their identity is confirmed in AD. All of these parameters can be changed later.

Now you are redirected to the overview page for your registration. Congrats! It exists.

Configuration for the CTF

Navigate to the registration overview page. In the left-hand sidebar, under manage, select Authentication. In the Web panel, click Add URI. Then, in the box that appears, type `https://ctf-domain-here/usersetup.html` and save. It is required that the URI be `https`, unless it is a `127.0.0.1` or `localhost` address. If development is still ongoing, add

the URI `http://localhost:4000/usersetup.html` as well for ease of testing.

Scroll down to the Implicit Grant and Hybrid Flows heading. Check both boxes to enable both Access Tokens and ID Tokens. Then, double check under Supported Account Types that only accounts in WPI's organization can access the application.

Note:

This configuration will cause Azure to display warnings which suggest that the implicit flow is not supported. However, this only applies to applications using Microsoft's own JavaScript library to handle authentication requests. We are using a different library which does support implicit flows. We do not need to migrate URIs, though our authentication will still work if you choose to do so.

Updating the Application Code

Finally, we need to update the website's code. Since we want to associate it with a new app registration, we need to make adjustments to some parameters of our authentication requests.

In the `pwnable-claw-machine project`, navigate to `picoCTF-web/web/jsx`. We will be updating two files in this directory.

First, go to `front-page.jsx`. Find the function called `startSignInMainWindow()`, which should be close to the beginning of the file.

In this function, we are going to be updating the settings object field-by-field.

The `authority` parameter is already configured correctly for WPI's Active Directory. If you wanted to change it to a different AD, then you would update the hex-string portion of the URL to be the Tenant ID of the new AD, which can be found on the application registration overview.

The `client_id` needs to be the Application ID of your application registration. This can also be found on the overview page of your app registration.

Finally, make sure that the `redirect_uri` and the `post_logout_redirect_uri` fields accurately reflect the URL of your website, and that they begin with `https`. The `redirect_uri` field must be exactly the same as one of the redirect URIs specified in your application registration, and the `post_logout_redirect_uri` should be the front page of the site. All other fields of the settings object remain the same.

Copy the entire settings object to your clipboard.

Now, navigate to the file `usersetup.jsx`, and find the function `endSigninMainWindow()`. This function also contains a settings object. Delete it, and replace it with the copied object. The settings in both files should be identical. Save both files, and your configuration is done!

Deploy Changes

Finally, we must ensure that our updated code is pushed to the CTF website. This process varies depending on whether you currently have the CTF built.

If you re-run the VM setup process with `vagrant up --provision`, your changes will be incorporated. However, if you do not want to rebuild the entire CTF, it is possible to use Ansible to update only the necessary components of the CTF.

First, navigate to `pwnable-claw-machine/infra_local` in your terminal. Then, run the command `ansible-playbook site.yml --limit web --tags web-static`, which will rebuild the website and incorporate your changes.

If you do not have Ansible installed on your local machine, you can use `vagrant ssh web` to access the virtual machine running the website. Then, you can follow the same steps as above to rerun the Ansible playbook.

The website will now reflect your changes.

Note:

If you are not running the full CTF, or if you would like to test the website without having to provision several VMs, see Appendix I for detailed steps on locally building and serving the website without the backend API.

References

[1] AzureAD. Azuread/microsoft-authentication-library-for-js: Microsoft authentication library (msal) for js.

[2] David Capello. Asesprite.

[3] Julian Descottes. Piskel: Free online sprite editor.

[4] Auth0 Docs. Implicit flow with form post.

[5] Auth0 Docs. Implicit flow with oidc.

[6] OAuth 2.0 Documentation. Id tokens vs access tokens.

[7] OAuth 2.0 Documentation. What are oauth access tokens.

[8] OpenID Foundation. Openid connect, Nov 2022.

[9] Google. Material design dark theme.

[10] IdentityModel. Identitymodel/oidc-client-js: Openid connect (oidc) and oauth2 protocol support for browser-based javascript applications.

[11] Rohit Jha. Introduction to 'capture the flags' in cybersecurity, Jun 2020.

[12] Mozilla Developer Network. Getting started with react.

[13] Scott Neuman. Grumpy cat awarded $710,000 in copyright infringement suit, Jan 2018.

[14] Stack Overflow. Stack overflow developer survey 2022.

[15] Katie Van Syckle. Keyboard cat and nyan cat come out ahead in lawsuit against warner bros.., Sep 2013.

[16] Carnegie Mellon University. Picoctf.

[17] Carnegie Mellon University. Picoctf/picoctf: The platform used to run picoctf 2019.