CISC472 PENTESTING REPORT

by

Calvin Banning

A self-critique of the UX and security for the final project of CISC472. Hosted versions of the project can be found at https://banningcalvin-redchan.glitch.me/ and at https://github.com/banningcalvin/redchan/.

2020

TABLE OF CONTENTS

Chapter 1

SECURITY EVALUATION

A security evaluation for the final project of CISC472, evaluating the group-project of Cong Meng and Owen Li. Hosted versions of the project can be foundat diy-reddit-257012.web.app and at https://github.com/udcymen/diy-reddit.

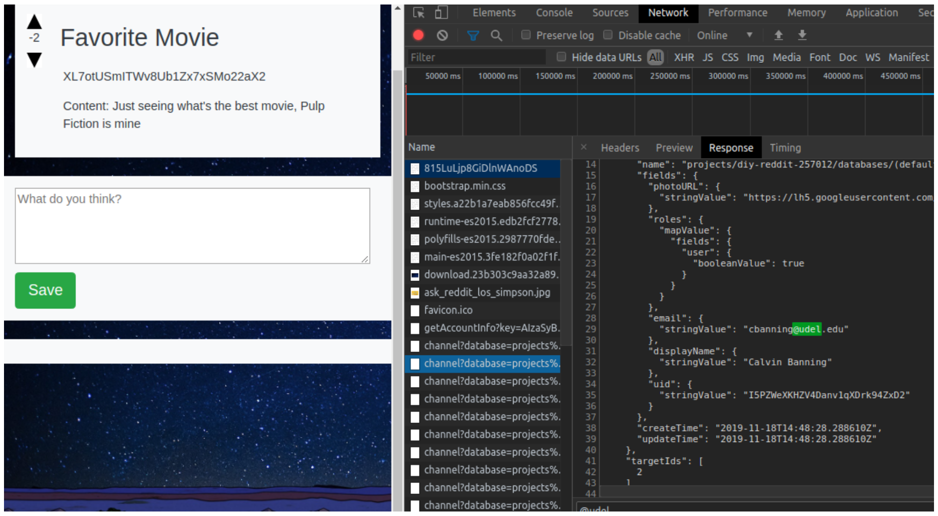
1.1 Trivial Vulnerabilities

These vulnerabilities will not result in data loss, but may result in an adverse user experience. For this project, it is completely understandable that these would be present for testing purposes, but a production environment should have solutions to all of these.

- Users can comment very quickly and overload a post with comments.
- Users can create accounts quickly and there is no account verification.
- Because of these above two vulnerabilities, it is conceivable that the user could write a script to overload the site with requests (as there appears to be no DDOS protection). Furthermore, the user could use this script to make requests to make an account and vote or post to spam the site with bogus votes or posts from a single real user. This could have massive financial implications for the developers if they were in a production environment that was configured to scale with user activity.
- User UIDs are displayed openly.
- The Firebase SDK is running in developer mode.

1.2 Data Vulnerabilities

In addition to UIDs being openly displayed, examining network output reveals even more data:



-2
Favorite Movie

XL7otUSmITWv8Ub1Zx7xSMo22aX2

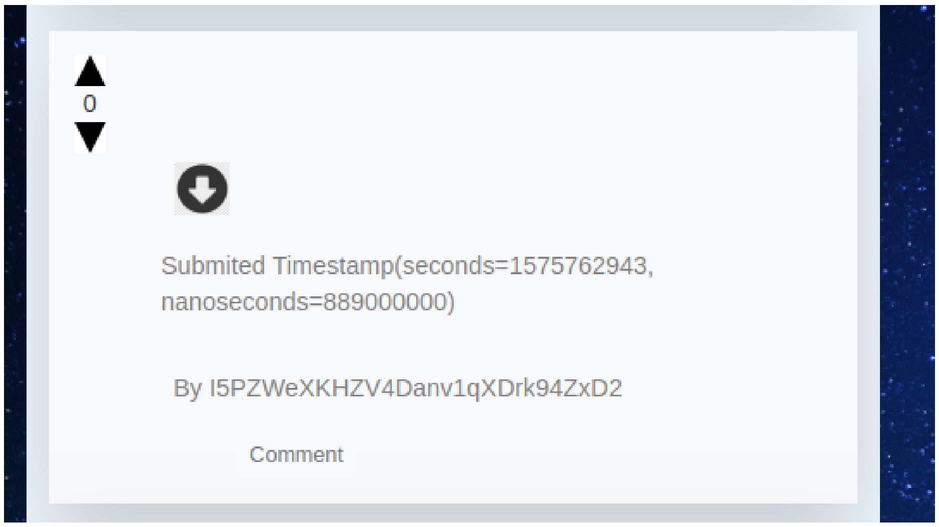
Content: Just seeing what's the best movie, Pulp Fiction is mine

What do you think?

```
        "name": "projects/diy-reddit-257012/databases/(defaul
        "fields": {
          "photoURL": {
            "stringValue": "https://lh5.googleusercontent.com
          },
          "roles": {
            "mapValue": {
              "fields": {
                "user": {
                  "booleanValue": true
                }
              }
            }
          },
          "email": {
            "stringValue": "cbanning@udel.edu"
          },
          "displayName": {
            "stringValue": "Calvin Banning"
          },
          "uid": {
            "stringValue": "I5PZWeXKHZV4Danv1qXDrk94ZxD2"
          }
        },
        "createTime": "2019-11-18T14:48:28.288610Z",
        "updateTime": "2019-11-18T14:48:28.288610Z"
      },
      "targetIds": [
        2
      ]
```

The personal email, name, and UID of other users can be easily seen if they posted on the page you are viewing by examining network responeses.

The single largest vulnerability I found related to making new posts:



0

Submited Timestamp(seconds=1575762943, nanoseconds=889000000)

By I5PZWeXKHZV4Danv1qXDrk94ZxD2

Comment

The form disables the submit button until you fill out all fields. However, the html can be edited to enable it, so one can submit a post without these fields (resulting in empty posts). I also conceived of another attack in which the antagonist could host their own version of the site and edit the typescript model for posts to include invisible fields, then post these to an area of the db where they had write access, effectively using posts as free databases.

The developers also gave me access to their frontend code repository. In this, we can see that the angular service for creating posts takes in a UID from the user, rather than generating it based on the auth token. Because of this, the user can pass in a different UID, making posts seem as if they come from a different author. See below for the relevant code:

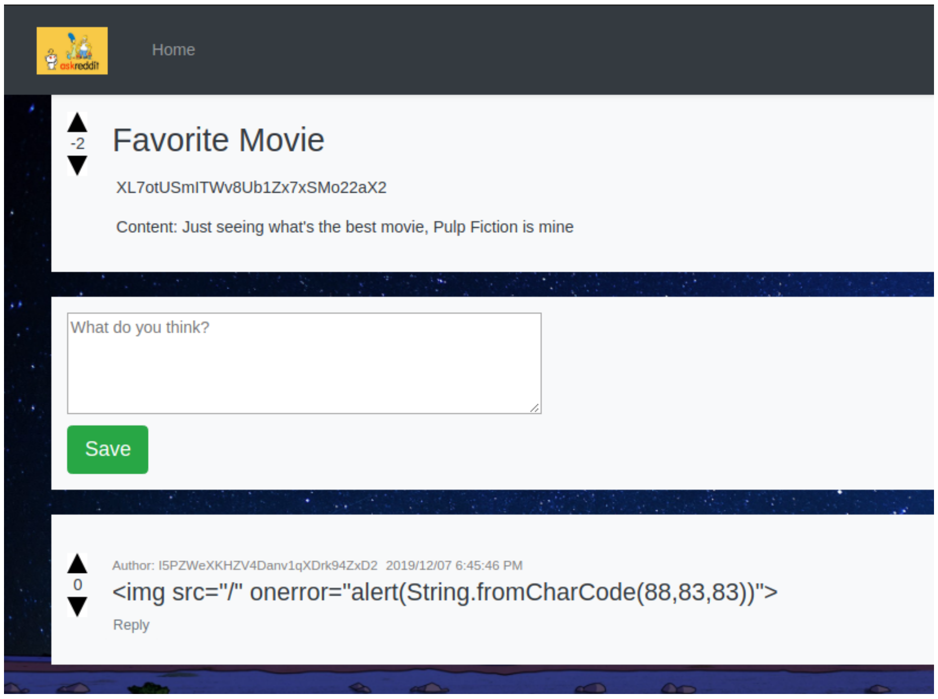
```
/*
 * src/app/views/create-post/create-post.component.ts
 */
this.postService.addPost(this.topic,
                         this.title,
                         this.content,
                         this.user.uid)
```

Simply put, this.user.uid should be implied from the auth token and the user should not be trusted to be honest about their identity.

1.3 Testing of other vulnerabilities

I briefly tried a variety of common attacks to see if the application was vulnerable:

- Firebase is not vulnerable to SQL injection, and it appears that all data is sent and received from firebase.
- Examining network output does not seem to reveal any data or other information that would indicate a XSS (cross-site-scripting vulnerability). See below for one test:



-2
Favorite Movie

XL7otUSmITWv8Ub1Zx7xSMo22aX2

Content: Just seeing what's the best movie, Pulp Fiction is mine

What do you think?

Author: I5PZWeXKHZV4Danv1qXDrk94ZxD2 2019/12/07 6:45:46 PM

0
<img src="/" onerror="alert(String.fromCharCode(88,83,83))">

- Session management appears to be solid. All of this seems to be followed according to firebase specifications, and trying to break this would be a futile exercise.

1.4 Vulnerability Resolution

The most important vulnerability to patch is how posts are attributed to authors. The developers must stop attribute posts based on auth tokens, rather than a user-supplied UID. Having a Firebase function which assigns the author value in the database rather than allowing the user to pass in an arbitrary value is a quick and secure fix for this.

Next, various steps should be taken to secure this development environment before it moves to production. Identify users with a username rather than a UID, and prevent information like their personal email from being leaked by comment and post data.

Finally, spam attacks can be prevented by adding post timers or a captcha to all registrations, posts and comments, which would prevent users from creating massive volumes of phony accounts, comments, posts, and votes.

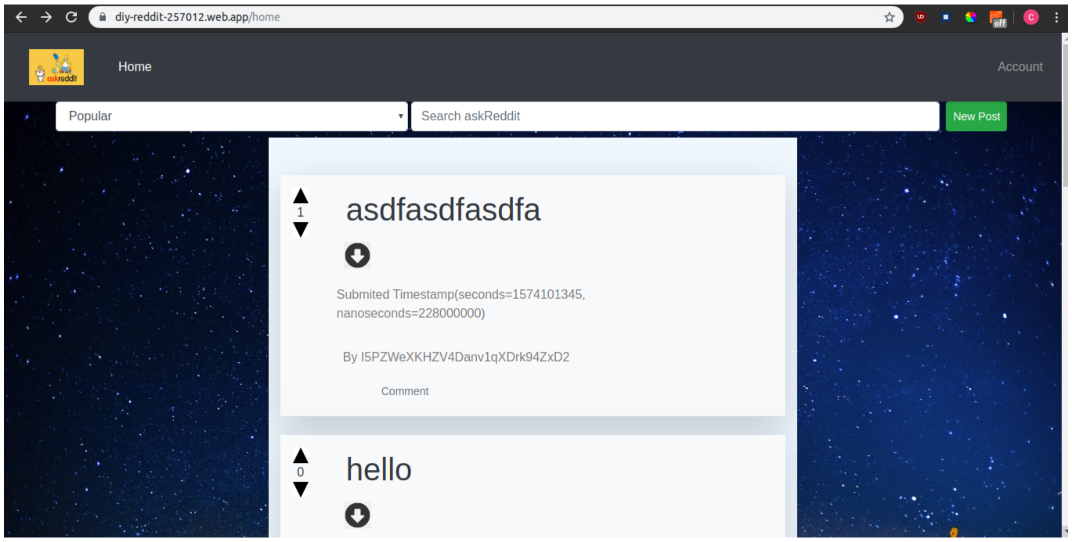
Chapter 2

UX QUALITY ASSURANCE

A design evaluation for the final project of CISC472, evaluating the groupproject of Cong Meng and Owen Li. Hosted versions of the project can be foundat diy-reddit-257012.web.app and at https://github.com/udcymen/diy-reddit.

2.1 Overview

Below is a screenshot of the home page. Ignoring post content, which is understandably meaningless because of numerous test posts, we can still see several flaws.



asdfasdfasdfa

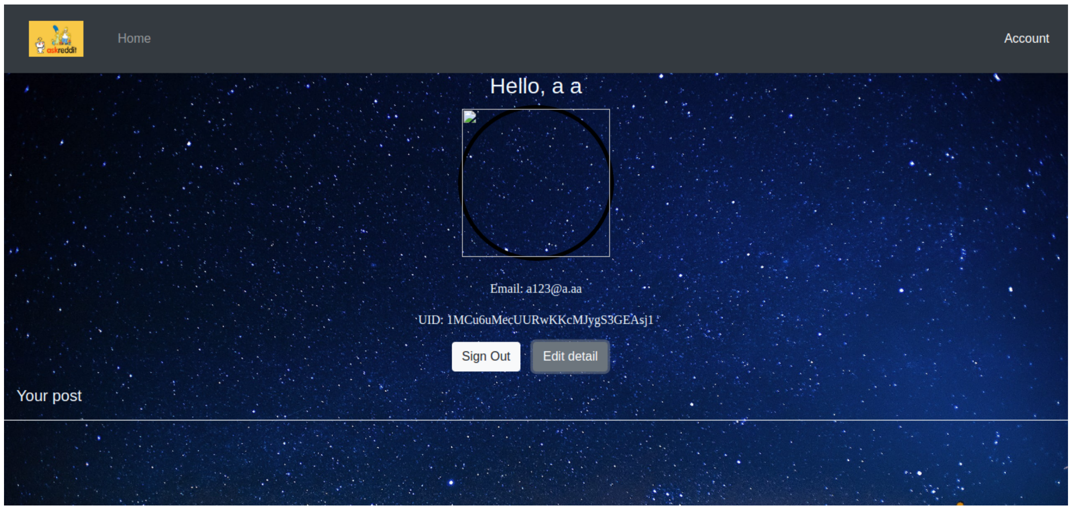
Submited Timestamp(seconds=1574101345, nanoseconds=228000000)

By I5PZWeXKHZV4Danv1qXDrk94ZxD2

hello

- There is an incredible volume of useless or confusing information. Timestamps are given in nanoseconds and seconds, but not as dates or another, more easily consumable format.
- Posts are not ordered correctly by genre or other filters. Furthermore, the 'subreddit' chooser does nothing. The dropdown isn't the most user-friendly choice, and any tab shows the exact same posts, and fails to show more than about 10. After this, no new posts can be seen. Even posts with negative votes appear above many of those with positive counts.

- When creating a new post, you are given a rather well designed popup that alerts you of the post ID (a link or simple alert without a long id string might have been better). However, as previously mentioned, there is no way to see this post. Additionally, navigating to the Account page does not correctly list posts.
- Navigating between pages feels jittery. Rather than waiting to render the page until a response has been received, the page renders, and then it is populated after post data is received from the backend.
- There is little consistency in design. Default fonts are used sporadically, with confusing mixes of serif and sans-serif. The logo and background contain characters from The Simpsons.
- The Account Page has some useless buttons (see below), While the buttons are well styled, they are inconsistent (see the plaintext home link and the well-styled Sign Out Button).



Hello, a a

Email: a123@a.aa

UID: 1MCu6uMecUURwKKcMJygS3GEAsj1

- The forgot password button does not work.
- The individual post view is much better, as it shows comment dates. However, it still displays poster IDs, which is also a pontential security concern. Additionally, the content is prefixed by the words 'Content' in the exact same font as the content, which is unneccesary and confusing.
- I found some elements to be too large. Even though my laptop screen is 1376*768px, it still felt like I was browsing on a mobile phone.
- The mobile version of the site is not functional. Buttons are hidden and the site is not entirely responsive.

Put simply, this project feels unfinished. understandably, it will not be fully featured, but it should be expected that features present should be fully fleshed out.

2.2 Evaluation

The website as it stands is still fairly simple, but it is almost functional enough for a production environment from a UX perspective. The single largest shortcoming is the inability to see new posts. After than, it is the useless functionality of the subreddit selector. Fixing these elements would effectively make the product MVP-ready.

2.3 Suggestions

Spending an hour cleaning up vestigal controls, make all styling uniform, and fixing spacing, sizing, and overflow rules would solve 50% of issues. Additionally, the issues with timestamps being displayed in nanoseconds should be a simple fix, considering it was done correctly in other areas of the site. Removing IDs and replacing them with a username would also be a good idea.

The remainder of shortcomings result from unexpected behavior. Adding tags to posts that allow the subreddit selector to correctly filter would add tremendous amount of depth to the user experience. Fixing the inability to see new posts, and incorrect sorting of posts based on votes would also fix most of the user experience.

Finally, adding more to the site in general would be useful: Descriptions, a solid background (the current background is distracting), and sidebars, would drive user interaction and reduce the feeling of an 'empty' application.