Jarrin Kasuya

Olivia Murray

Collin Wong

ICS 427

Endicott-Poposky

April 26, 2020

Four-Sided Triangle Report

Introduction

Team Name

Four-Sided Triangle

Team Members

1. Jarrin Kasuya
2. Olivia Murray
3. Collin Wong

Application Title, Description, and Functional Requirements Specification

"Never Alone"

- allow users to create their own study groups via our application
- they have the ability to view other study groups
- users can only delete their own study groups and are allowed to join others
- users can create their own profiles to match other student's preferences

Type of Program

Web Application

Development Tools

- Visual Studio Code, Notepad++
- Language: JavaScript

<u>**Requirements**</u>

<u>Security and Privacy Requirements</u>

Security Plan:

Before the start of our development, we need to create an outline that shows overall protection for our app. This includes initial development plans such as hosting services to maintenance issues where we deal with bug catching. This outline will be included in our Github repository that is currently private. The three of us will have access to the repository and we can see what kind of code is being pushed up the branches. There will also be an inventory check of the resources used. This is to ensure that we as developers are using the correct and safe libraries that are free from vulnerabilities.

Privacy Options for Users:

We do not plan on tracking user data. The only available data that a user is able to provide is their name and their academic background at UH Manoa. We would like to use cookies as little as possible to minimize the risk of user data being attacked or stolen. General requirements that we have include encryption for user credentials and data. For web traffic, we plan on implementing HTTPS for more secure browsing. This builds up our credibility from our users that we value their time, data, and privacy.

Identifying Security Flaws

Within our Github repository, we will have many different branches for each task that a developer will have to fulfill. There will be one master branch, one deployment testing branch, and all the other task assigned branches. When a developer has finished a task, they can push their branch to the deployment branch to check for security compatibility. With this general Github system, if a developer were to accidentally push a branch with a developer API key, that information will be on the deployment test branch and not immediately pushed to the master branch which would then be connected to our main hosting service. Github will also notify us of this security threat if we, the developers, did not see this flaw at first.

Quality Gates

Privacy Gates for Users

Critical

- Collection of User Data Without Consent
    - Example: We are collecting user data through cookies, information that is stored within cookies are also confidential information such as user authentication. There is also no encryption to protect this type of data.
- Age of Consent Usage
    - Example: Users who are not affiliated with UH Manoa being able to create an account to meet random people online.

Moderate

- Collection of User Data Without Consent
    - Example: We are collecting user data where there is an opt-out function. The opt-in function is turned on by default but the user has the necessary power to turn it off if they wanted to at their convenience.

Low

- Collection of User Data Without Consent
    - Example: Our web application collecting user data locally using cookies, however the data that is being stored is encrypted and is not accessible to the public.

Security Gates for Users

Critical

- Unwanted Privilege Promotions
    - Example: Users are able to execute SQL injections within our web application to change their role from user to admin as they are able to change their privileges by modifying database data.

Moderate

- Misrepresenting/Catfishing Users
    - Example: Users being able to create accounts that are not under their own name, creating a fake profile and making false study groups. This ruining other people's time and taking space on the study groups page.

- Denial of Service
    - Example: Attackers creating a botnet to constantly create new users as well as new study groups consuming all the resources needed on the web application. This slows the application down for everyone as there should not be the constant use of the same pages being open and created at the same time.

Risk Assessment Plan For Security and Privacy

To assess our app for security and privacy, we will follow a general template, provided by Microsoft. We will check to see if any component of the application performs any of the behaviors listed below:

- Storing/Transferring Personally Identifiable Information (PII) to/from the user's computer.
- It provides an application with a target audience of children.
- Continuously monitors the usage of the application.
- Installs new software or changes user preferences.
- Transfers anonymous data.

The above behaviors do not apply to this application. This application does not plan to store any personally identifiable information to a person, besides their name (necessary for study groups). This application's intended audience is students. This application will not monitor any user of the application and will not ask the user to install any software. This application will not track user data, so there is no reason for anonymous data transfer.

In the current state, one part of the application that should undergo a security/privacy review is the part of the application that requires the storage of the user's name and academic information. This is the only PII that the application deals with, and thus, it should undergo a serious security review to make sure that this information is not mishandled by the application in any way. Another part of the application that should undergo a security/privacy review is the fact that anyone can create an account, and therefore, while the target audience of the application is not intended to be children, accounts should be reviewed to be sure of this.

Design

Design Requirements

This application should be a simple one with an emphasis on a very simple user interface and easy accessibility. Another emphasis for this application is one that does not track user data, hence, there will be an emphasis on security for much of the design requirements. Here is a list of some design requirements for this application:

- The intended function is for users to create study groups or join other already created study groups. Therefore, there should only be two options on the homepage of the application: Create a Study Group and Browse Study Groups. The homepage should both describe the application, and allow a user to create a profile or log in.
- Implementation of HTTPS certificate. One of the main goals of this application is to not track user data in the application and to provide a secure environment. This will also help to ensure encryption for any user credentials and data.
- For the creation of user profiles, it should be done with a trusted certificate to verify encryption for any kind of user data.
- Users should be able to customize their profiles to change any academic preferences, background, etc. Also on this page, it should display all the study groups created by the user, giving them the option to delete any old study groups. This should also be done with a trusted certificate to prevent tracking any kind of user data.
- Users should also be able to browse or filter study groups based on their academic background or preferences. For example, if a user wants to find a study group to help with Chemistry, there should be a feature to sort for study groups dedicated to Chemistry. As we do not want to track user data, this should be done with encryption as well.

Attack Surface Analysis and Reduction

There will be three privilege levels for this application. The first level is available to all who visit this web application. There is no sign in required. This will display the base of our web application and will discuss the features of our web application to try and get those who are not already signed up to join.

There is then a privileged user level where one must first create an account which will require an email address and a password for future logins. Here, users will be able to view their customized web application that displays information for that specific user.

The final level is the executive level where the managers of the page will be able to view managerial information about the users. This is an easy interface to be able to view information about the user and make some changes to database entries.

Some points of entry into this web application include:

- User interface (UI) forms and fields
- HTTP headers and cookies
- APIs
- Files
- Databases
- Other local storage
- Email or other types of messages
- Run-time arguments

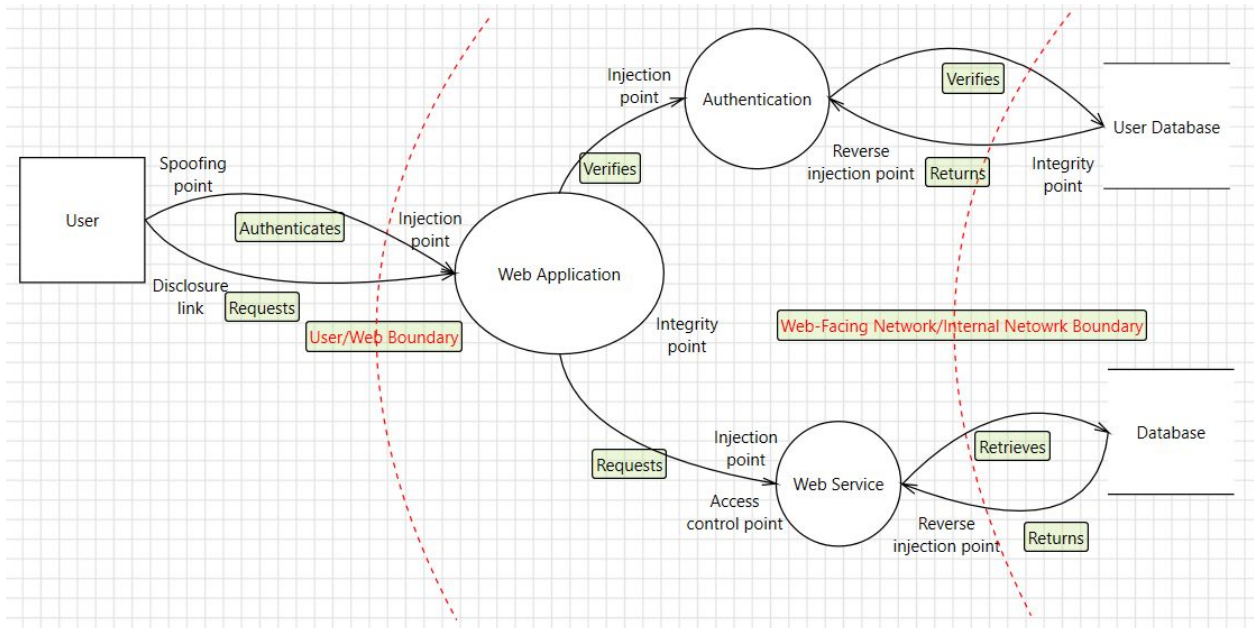
Some motivations for hackers to try and attack this web application include private user information such as email, password, and phone number.

The most common web application security vulnerabilities are as followed:

- SQL Injection
  - Allows an attacker to alter backend SQL statements by manipulating user-supplied data

- Cross-Site Scripting
    - Target scripts embedded in a page that is executed on the client-side i.e. user browser rather than at the server's side
- Broken Authentication and Session Management
    - Often occurs when a user is using a public computer and sensitive data still exists in the system after the user leaves the computer
- Insecure Direct Object References
    - Occurs when a developer exposes a reference to an internal implementation object such as a file, directory, or database key and an attacker uses this information to access other objects and create future attacks to access unauthorized data
- Security Misconfiguration
    - Security configuration is not correctly defined an deployed, allowing an attacker to have unauthorized access to sensitive data or functionality.

Threat Modeling



Threats by Category

Spoofing

- Spoofing the Human User External Entity
- Spoofing the Web Application Process
- Spoofing of Destination Data Store Generic Data Store
- Spoofing of Destination Data Store User Database
- Spoofing of Source Data Store User Database
- Spoofing the Web Service Process

Tampering

- Cross-Site Scripting
- Potential Lack of Input Validation for Web Application
- The Database Data Store Could Be Corrupted

Repudiation

- Potential Data Repudiation by Web Application
- Data Store Denies Database Potentially Writing Data

- Potential Data Repudiation by Web Service
- Misrepresenting/catfishing users
- Age of consent usage

Information Disclosure

- Data Flow Sniffing
- Weak Access Control for a Resource
- Collecting user data without consent

Denial of Service

- Potential Process Crash or Stop for Web Application
- Data Flow Requests Is Potentially Interrupted
- Potential Excessive Resource Consumption for Web Service or Generic Data Store
- Potential Excessive Resource Consumption for Authentication or User Database
- Data Flow Retrieves Is Potentially Interrupted
- Any type of DoS attack
- Creation of fake profiles

Elevation of Privilege

- Elevation Using Impersonation
- Web Application May be Subject to Elevation of Privilege Using Remote Code Execution
- Elevation by Changing the Execution Flow in Web Application
- Elevation Using Impersonation
- Insecure direct object references
- Security misconfigurations

Implementation

Approved Tools:

*** Note all tools and resources are used for Windows 10*

JavaScript Tools

Name	**Version**	**Comments**
Node.js	v12.14.1	https://nodejs.org/en/
Chocolatey	v0.10.15	https://chocolatey.org/

JavaScript Libraries/Frameworks

Name	**Version**	**Comments**
Meteor	v1.9	https://www.meteor.com/
React	v16.12.0	https://reactjs.org/
react-dom	v16.12.0	https://reactjs.org/docs/react-dom.html
react-router-dom	v5.1.2	https://www.npmjs.com/package/react-router-dom
react-scripts	v3.3.1	https://www.npmjs.com/package/react-scripts
semantic-UI-react	v0.88.2	https://react.semantic-ui.com/

Code Editors

Name	**Version**	**Comments**
Visual Studio Code	v1.42	https://code.visualstudio.com/
Notepad++	v7.8.4	https://notepad-plus-plus.org/

Deprecated/Unsafe Functions

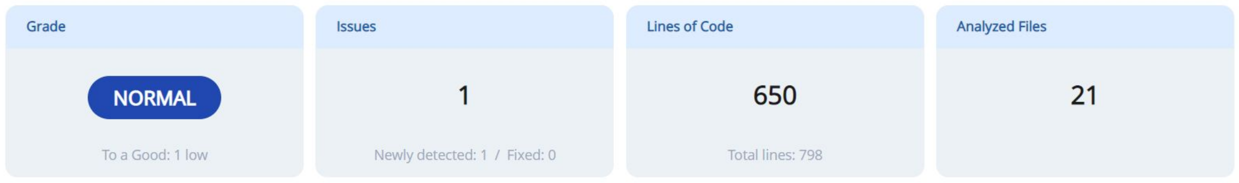
- Node.js
    - eval()
        - Can allow attackers to get full access to the production environment
        - Alternatives:
            - Use JSON.parse()
            - Validate input with Joi
- Meteor
    - meteor create myApp
        - Meteor packages created include autopublish and insecure
        - autopublish takes the server-side database and mirror of all its contents to the client
        - insecure will grant any connected client to the rights to modify any data in the database as if they had server access
        - Alternatives:
            - Can use this function/command but after, delete packages autopublish and insecure
- React
    - JSON.stringify()
        - Will blindly turn any data it is given into a string which will be rendered in the page so fields that untrusted users can edit can have other codes injected
        - Alternatives:
            - Use the serialize-javascript NPM module to escape the rendered JSON
    - Sneaky Links using javascript:
        - An attacker can add a payload prefixed with javascript:
        - Alternatives:
            - Delete javascript: prefix
            - Add additional protection to make sure user links open in new tabs
            - Use a special UserLink component
    - dangerouslySetInnerHTML
        - The name explains this one but it dangerously sets an inner HTML
        - Alternatives:
            - Make sure the instance of this method only loads trusted data

Static Analysis Tool: DeepScan

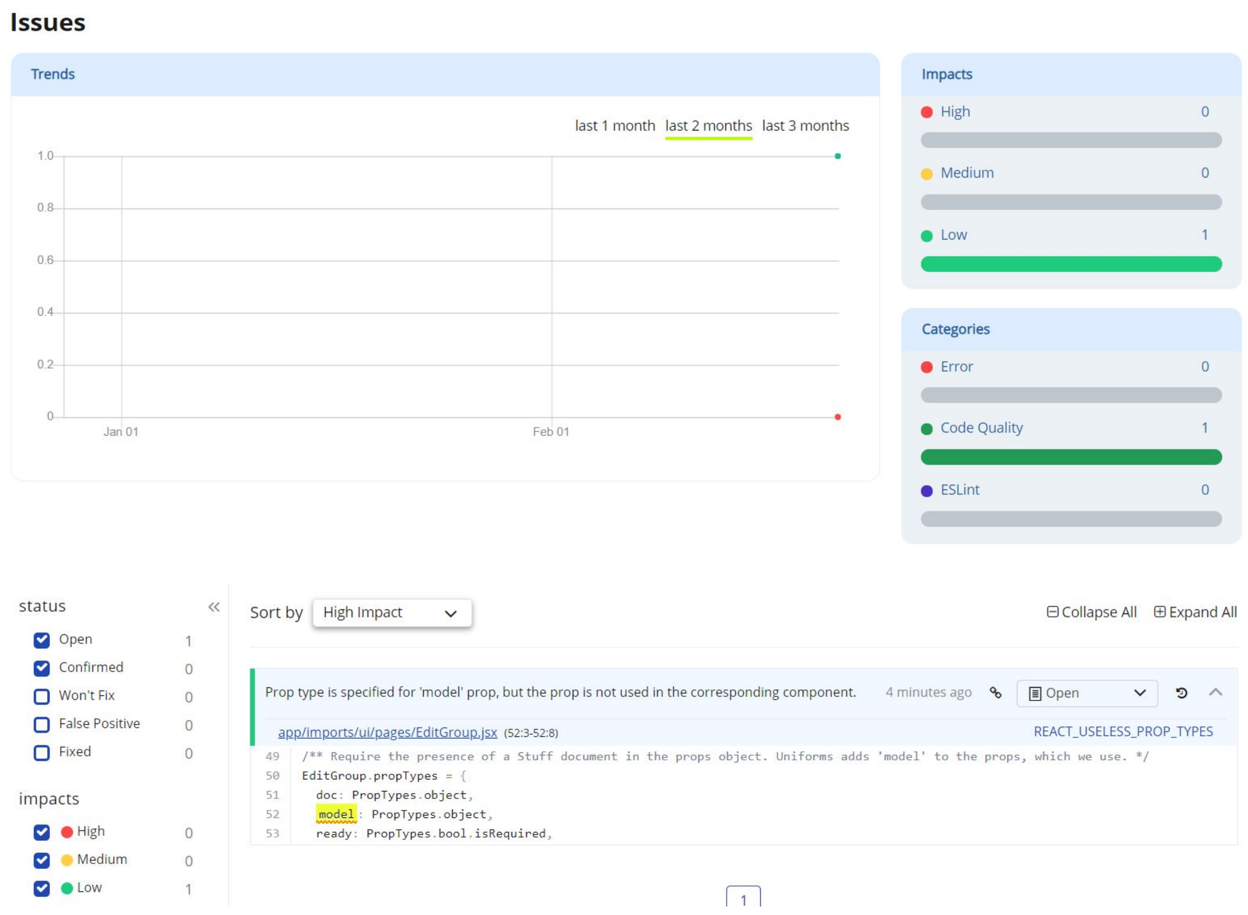
The static analysis tool we have chosen to use for this project is DeepScan. DeepScan is a static analysis tool for JavaScript code that helps to ensure code quality. DeepScan works in a similar fashion to ESLint, as it helps to point out any errors in the code, but also gives a more detailed analysis than ESLint. DeepScan also helps to show how the code may be executed, such as giving warnings that a condition statement may always be true or false because of how the condition was previously defined, etc.

Another positive to the DeepScan application is that it connects right with a GitHub account. There is both a free version that supports Open Source repositories, as well as a premium version that supports private repositories. As commits are pushed to GitHub, DeepScan will analyze the code and give analysis on any code issues, quality status, etc. DeepScan also gives a grade for the project (either “Poor”, “Normal”, or “Good”), which is dependent on the number of issues and the impact those issues have on the program, in the hopes that a lower grade will motivate the developers to improve their code. Finally, as DeepScan continuously analyzes code as commits are made to a specific GitHub repository, it also allows developers to see how code issues have progressed over time. It will show both unresolved issues (and will keep showing them as more commits are made if they are not fixed), as well as issues that were fixed.

Deep Scan Statistics:



Grade	Issues	Lines of Code	Analyzed Files
NORMAL	1	650	21
To a Good: 1 low	Newly detected: 1 / Fixed: 0	Total lines: 798	



Impacts	
● High	0
● Medium	0
● Low	1

Categories	
● Error	0
● Code Quality	1
● ESLint	0

status	
✓ Open	1
✓ Confirmed	0
Won't Fix	0
False Positive	0
Fixed	0

impacts	
✓ ● High	0
✓ ● Medium	0
✓ ● Low	1

Prop type is specified for 'model' prop, but the prop is not used in the corresponding component. 4 minutes ago Open

app/imports/ui/pages/EditGroup.jsx (52:3-52:8) REACT_USELESS_PROP_TYPES

```
/** Require the presence of a Stuff document in the props object. Uniforms adds 'model' to the props, which we use. */
EditGroup.propTypes = {
  doc: PropTypes.object,
  model: PropTypes.object,
  ready: PropTypes.bool.isRequired,
```

After initially using DeepScan to analyze our project, it showed that there was only one low-impact issue in the project that could be easily fixed (in this case, using a prop type when it wasn't necessary). It showed that it was in the code quality category of static analysis, which is focused on improving code and making it more efficient. It gave our project a grade of "Normal", which could be easily improved to be "Good" after the issue was fixed. There were no difficulties involved in using this application.

Overall, using DeepScan as a static analysis tool helped us to see where and how we could improve the quality of our code to make things more efficient and less cluttered. We will continue to use DeepScan to analyze our code as we make progress in the project to ensure our code is at the highest possible quality without any errors.

Dynamic Analysis Tool: iroh.js

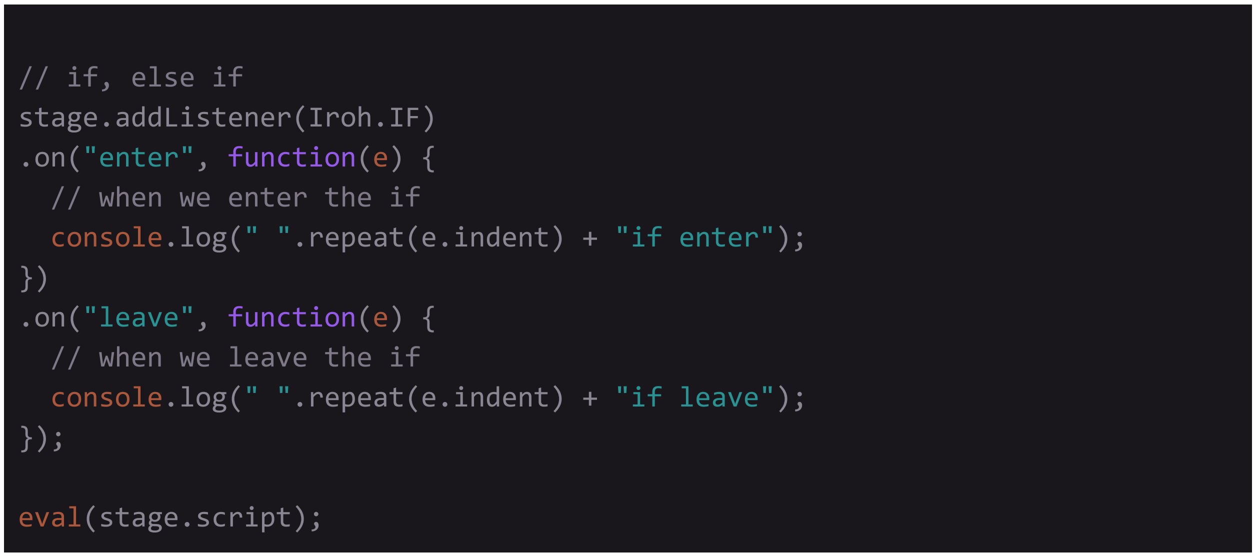
Website: https://maierfelix.github.io/Iroh/

Github: https://github.com/maierfelix/Iroh

The dynamic analysis tool that we have decided to use for our project is called iroh.js. This tool allows us to see how our application will behave when it is executed, by tracking, intercepting, and manipulating data during code execution. Iroh.js helps to record what happens to patched functions (segments of interest) or variables, showing how the code flows and is manipulated through runtime. This is helpful to visualize the results of the code, rather than running the program multiple times with different inputs to figure out how it will react.

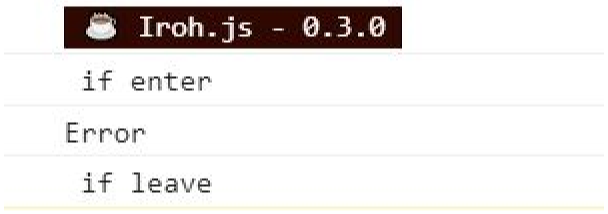
We tested this dynamic analysis tool on our application's adding permission form. The segment of code tested is shown below:

```
let code = `
 if (true) {
                  console.log('Error');
        } else {
        console.log("Success");
        }
`;

let stage = new Iroh.Stage(code);

// while, for etc.
stage.addListener(Iroh.LOOP)
.on("enter", function(e) {
  // when we enter the loop
  console.log(" ".repeat(e.indent) + "loop enter");
})
.on("leave", function(e) {
  // when we leave the loop
  console.log(" ".repeat(e.indent) + "loop leave");
});
```



```
// if, else if
stage.addListener(Iroh.IF)
.on("enter", function(e) {
  // when we enter the if
  console.log(" ".repeat(e.indent) + "if enter");
})
.on("leave", function(e) {
  // when we leave the if
  console.log(" ".repeat(e.indent) + "if leave");
});

eval(stage.script);
```

We then looked at what iroh.js told us the output to this particular segment would be.



```
Iroh.js - 0.3.0
  if enter
Error
  if leave
```

This showed that the code executed as we had expected. It helped us to visualize the runtime call stack to see what part of the code executed when. It showed that we entered the if statement, which logged “Error” into the console because the conditional statement was true, then showed we exited the if statement. We could easily have manipulated the data to show other cases, but this is a very simple example to show how iroh.js helps to record what happens in particular segments of the code.

At the time of writing, we have encountered no difficulties using iroh.js to dynamically analyze our code. We will also continue to use iroh.js to dynamically analyze our code, whenever changes are made, to verify that the application is still working as planned. Overall, using a tool for dynamic analysis is very helpful in the application development process to visualize how a segment of code will run without actually changing the program itself.

Attack Surface Review

Developer Tool Changes

JavaScript Tools

Name	**Version**	**Patch Notes**
Node.js	v12.14.1 → v12.16.1	https://nodejs.org/dist/latest-v12.x/docs/api/
Chocolatey	v0.10.15	No Change

JavaScript Libraries/Frameworks

Name	**Version**	**Patch Notes**
Meteor	v1.9 → v1.10.0	https://docs.meteor.com/changelog.html
React	v16.12.0 → v16.13.0	https://github.com/facebook/react/blob/master/CHANGELOG.md#16130-february-26-2020
react-dom	v16.12.0 → v16.13.0	https://www.npmjs.com/package/react-dom
react-router-dom	v5.1.2	https://www.npmjs.com/package/react-router-dom
react-scripts	v3.3.1 → 3.4.0	https://www.npmjs.com/package/react-scripts
semantic-UI-react	v0.88.2	No Change

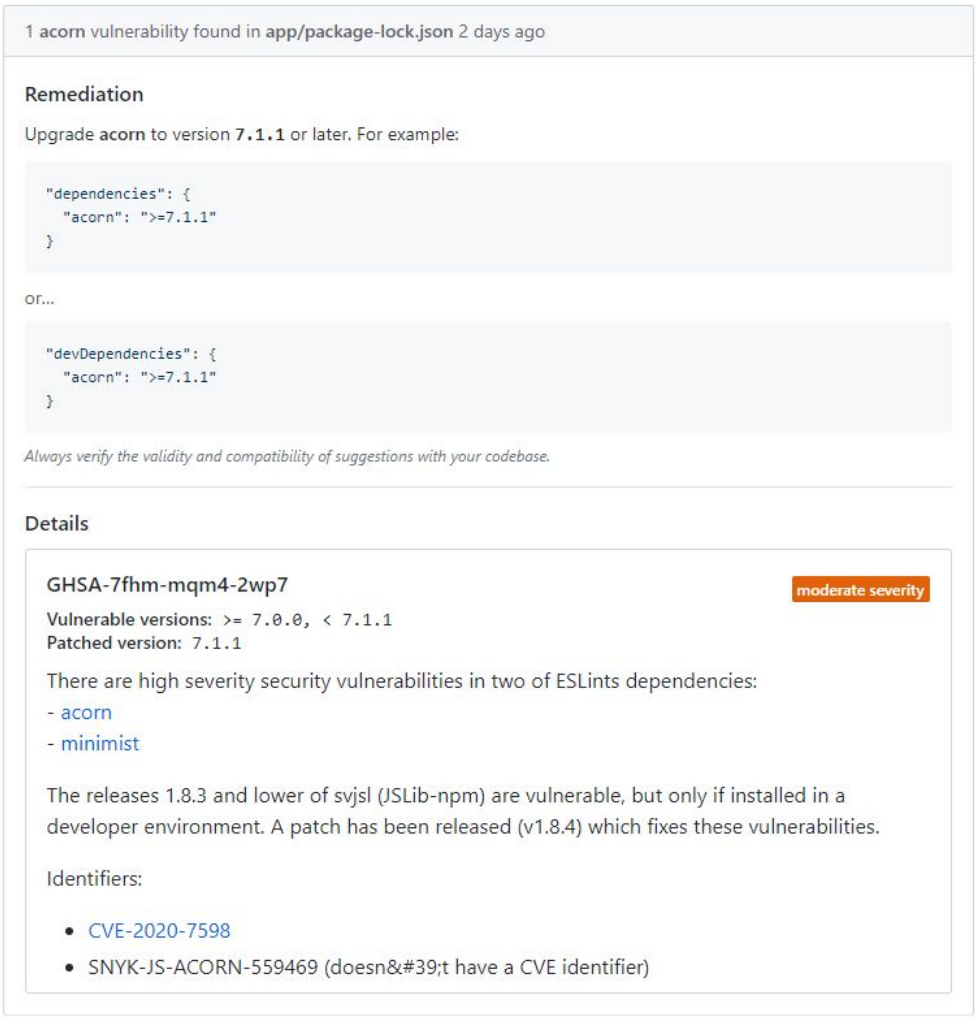
Code Editors

Name	**Version**	**Patch Notes**
Visual Studio Code	v1.42 → v1.43	https://code.visualstudio.com/updates/v1_43
Notepad++	v7.8.4 → 7.8.5	https://notepad-plus-plus.org/

	downloads/v7.8.5/

Going over the development notes for each of the tools that the group used, there were no significant changes in any of these tools besides minor updates. No vulnerabilities were reported, as most of these updates were just bug fixes. For certain libraries that were used such as semantic UI-react, there was no update at all since December 2019. The biggest change was with regards to Meteor v1.10.0 and the compatibility with MongoDB. This is because Cordova was updated from version 7 to version 9 which means certain plugins are incompatible. Also, MongoDB is no longer supporting 32-bit systems, only 64-bit systems.

There was a vulnerability found in two of ESLints dependencies for versions between 7.0.0 and 7.1.1. Below describes the vulnerability and remediation.



1 **acorn** vulnerability found in **app/package-lock.json** 2 days ago

Remediation

Upgrade **acorn** to version `7.1.1` or later. For example:

```
"dependencies": {
  "acorn": ">=7.1.1"
}
```

or...

```
"devDependencies": {
  "acorn": ">=7.1.1"
}
```

Always verify the validity and compatibility of suggestions with your codebase.

Details

GHSA-7fhm-mqm4-2wp7

moderate severity

Vulnerable versions: `>= 7.0.0, < 7.1.1`
Patched version: `7.1.1`

There are high severity security vulnerabilities in two of ESLints dependencies:
- acorn
- minimist

The releases 1.8.3 and lower of svjsl (JSLib-npm) are vulnerable, but only if installed in a developer environment. A patch has been released (v1.8.4) which fixes these vulnerabilities.

Identifiers:

- CVE-2020-7598
- SNYK-JS-ACORN-559469 (doesn&#39;t have a CVE identifier)

Verification

Fuzz Testing

We tried several methods of fuzz testing. We used the program JBroFuzz for web protocol fuzzing tests. This program generates requests and puts them on the wire, and then records the corresponding responses received back. In one test, we tested for responses received for invalid HTTP versions. This is a minor fuzzing test, but it was our first to familiarize ourselves with the tool and to test the responses. By changing the versions to invalid HTTP versions, we received no headers but yet with a full HTML body so still a lot of data being sent. This does not prove any vulnerabilities are present, so no action was taken.

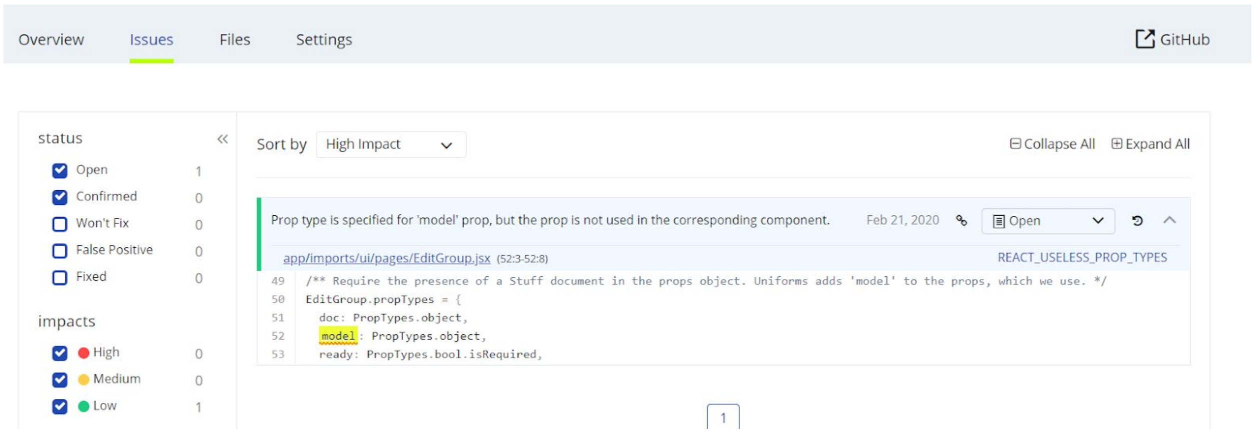
We then fuzzed a user ID. We transmitted 100 requests, changing the user ID each time. There was no distinguishing the difference between the response for an existing user and that of a user who does not have a corresponding ID. This does not prove any vulnerabilities in this area are present, so no action was taken.

Lastly, we tried SQL injections on any textbox that could be committed to POST. Trying different known SQL injections, we tried to give the application an SQL command, but none worked. There were other measures in place to prevent this from working. From this angle, there were no vulnerabilities to address.

There is obviously a lot more testing that can be done, but as of right now our application is in a fairly sturdy position. We will continue to test as we update the application and as patches are released for our application development tools.

Static Analysis Review

Deep Scan is a great tool that allows developers to analyze their Javascript code. Since the last assignment, there are some issues that were created and closed since then. In this example, it is a simple issue where a property is not being used. This does not affect the overall function of the EditGroup feature.

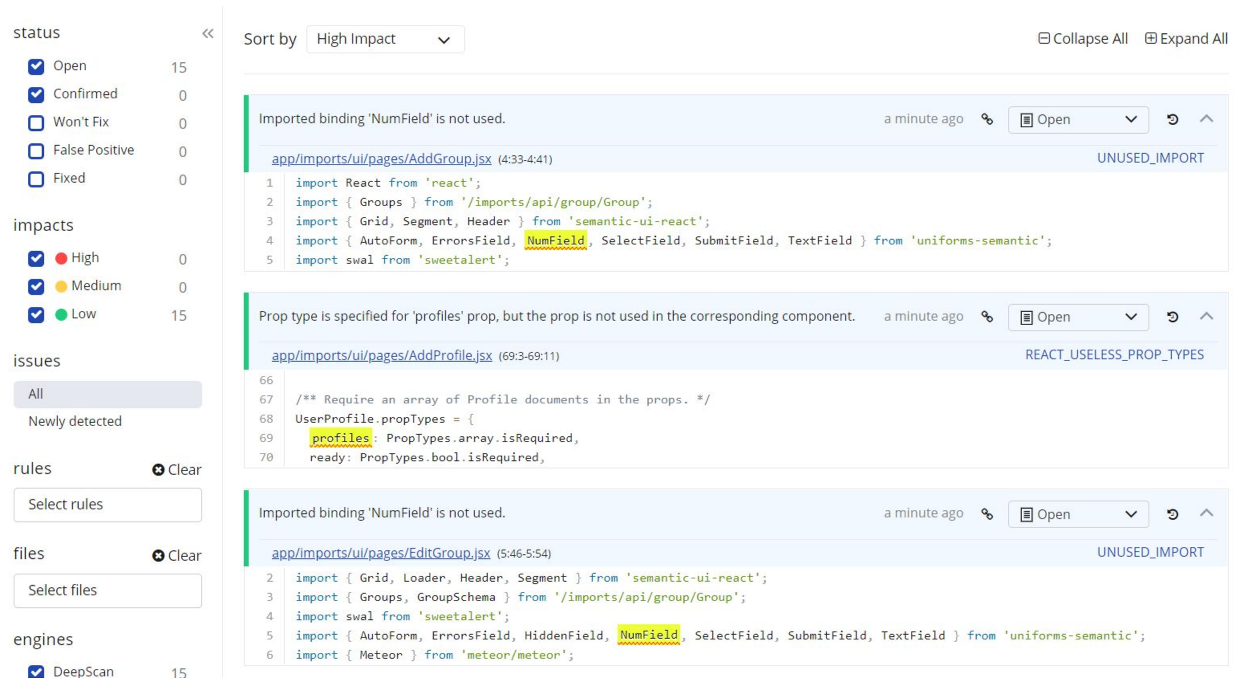


Prop type is specified for 'model' prop, but the prop is not used in the corresponding component.

app/imports/ui/pages/EditGroup.jsx (52:3-52:8) REACT_USELESS_PROP_TYPES

```
/** Require the presence of a Stuff document in the props object. Uniforms adds 'model' to the props, which we use. */
EditGroup.propTypes = {
  doc: PropTypes.object,
  model: PropTypes.object,
  ready: PropTypes.bool.isRequired,
```

There are also other new issues that have been opened that are also only on the "low impact" side.



Imported binding 'NumField' is not used.

app/imports/ui/pages/AddGroup.jsx (4:33-4:41) UNUSED_IMPORT

```
import React from 'react';
import { Groups } from '/imports/api/group/Group';
import { Grid, Segment, Header } from 'semantic-ui-react';
import { AutoForm, ErrorsField, NumField, SelectField, SubmitField, TextField } from 'uniforms-semantic';
import swal from 'sweetalert';
```

Prop type is specified for 'profiles' prop, but the prop is not used in the corresponding component.

app/imports/ui/pages/AddProfile.jsx (69:3-69:11) REACT_USELESS_PROP_TYPES

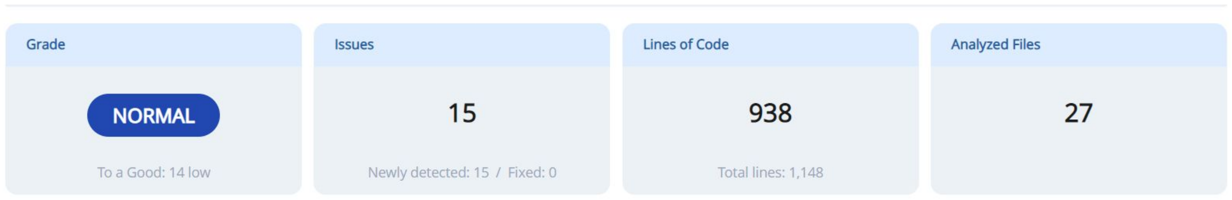
```

/** Require an array of Profile documents in the props. */
UserProfile.propTypes = {
  profiles: PropTypes.array.isRequired,
  ready: PropTypes.bool.isRequired,
```

Imported binding 'NumField' is not used.

app/imports/ui/pages/EditGroup.jsx (5:46-5:54) UNUSED_IMPORT

```
import { Grid, Loader, Header, Segment } from 'semantic-ui-react';
import { Groups, GroupSchema } from '/imports/api/group/Group';
import swal from 'sweetalert';
import { AutoForm, ErrorsField, HiddenField, NumField, SelectField, SubmitField, TextField } from 'uniforms-semantic';
import { Meteor } from 'meteor/meteor';
```

For the overall review of the deployment test branch, here are the statistics.



Grade	Issues	Lines of Code	Analyzed Files
NORMAL	15	938	27
To a Good: 14 low	Newly detected: 15 / Fixed: 0	Total lines: 1,148	

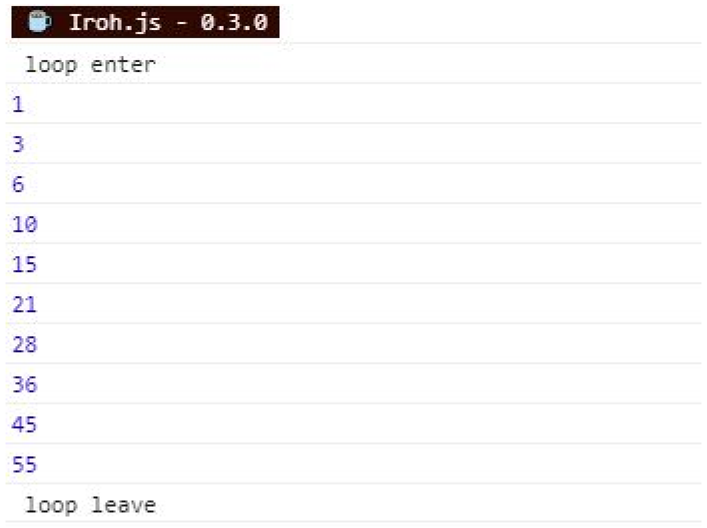
There have been new improvements and features to the applications, but although the number of issues may have increased, this does not change the core function of the app. These types of issues are meant as warnings that have no real effect on the main codebase.

Dynamic Analysis Review

The dynamic analysis tool our group chose to use for this project was Iroh.js. In the last assignment, the adding permission form was tested to make sure the application was running as expected. For this assignment, we decided to test another case of the same form. Instead of testing a conditional statement (“if” statements), we wanted to test a case for “for” statements. For simplicity, consider the following example below:

```
let code = `
 x = 0;
 for (i = 1; i <= 10; i++) {
   console.log(x += i);
 }
`;

let stage = new Iroh.Stage(code);

// while, for etc.
stage.addListener(Iroh.LOOP)
.on("enter", function(e) {
  // when we enter the loop
  console.log(" ".repeat(e.indent) + "loop enter");
})
.on("leave", function(e) {
  // when we leave the loop
  console.log(" ".repeat(e.indent) + "loop leave");
});

// if, else if
stage.addListener(Iroh.IF)
.on("enter", function(e) {
  // when we enter the if
  console.log(" ".repeat(e.indent) + "if enter");
})
.on("leave", function(e) {
  // when we leave the if
  console.log(" ".repeat(e.indent) + "if leave");
});

eval(stage.script);
```

To which we got the following output:



```
Iroh.js - 0.3.0
 loop enter
1
3
6
10
15
21
28
36
45
55
 loop leave
```

This again proves that our code is executing properly, as it successfully entered the loop, did what it was supposed to (added up the numbers 1 to 10), and then exited the loop. For a more practical example to our application, this would produce similar results if we were to add multiple permissions using a “for” statement.

Between the time the last assignment was written up until now, we have continued to use Iroh.js to visualize the call stack for function calls. We have not run into any major problems using it, nor have we noted any other big successes that helped us fix bugs in the code. We will continue to use Iroh.js in the future to verify our application is working as planned.

Release Report

Incident Response Plan

Privacy Escalation Team:

Escalation Manager: Olivia Murray

Oversees, coordinates, and prioritizes actions during evidence collection, analysis, and containment phases of incident response. The severity of the event will be analyzed and actions will be taken based on the severity and nature of the incident.

Legal Representative: Jarrin Kasuya

Provides legal advice during the incident response as well as representation to our team in the event that the incident develops into criminal charges.

Public Relations Representative: Olivia Murray

Communicates with external collaborators and the public. Any statements must go through her before being released and media outlets should be directed to her for questions.

Security Engineer: Collin Wong

Provides specialized technical knowledge, skills, and assistance during the incident identification, containment, and remediation. Maintains the incident response plan and updates security measures as needed.

Emergency Contact Email:

If there is an emergency, please contact Olivia Murray at omurray4@hawaii.edu.

Incident Response Procedures:

1. Identification and Scoping

    Incidents will be classified based on their perceived impact. Classifications may change as understanding of the incident improves. The first person to respond to the incident should attempt to give a first-estimate categorization in order to guide response. The escalation manager will make changes to this categorization from then out.

   Categories:

    High: An incident is considered High Severity if it involves a compromise of confidentiality or integrity of PII, a password database or store, software vulnerability information, attracts attention by media outlets or other public dissemination, is considered a major disruption to the application's ability to provide services, etc.

    Medium: An incident is considered Medium Severity if it involves a compromise of multiple accounts by the same party, a compromise that appears to be specifically targeted at our application, or a long-term disruption to non-critical services or degradation or critical services, etc.

    Low: An incident is considered Low Severity if it involves a disruption to a small number of users, a short-term disruption in our application's service availability (such as sue to a denial-of-service attack), an unsuccessful attempt to compromise our application's infrastructure in some way, etc.

2. Containment and Intelligence Development

    In any incident, it is important to preserve the evidence as well as act quickly to mitigate the impact of the event. In parallel with the formation of the privacy escalation team, the person doing the initial triage should take steps to prevent the problem from spreading to other accounts or resources, while being cautious not to raise the suspicions of the attacker.

3. Eradication and Remediation

   At this point, it is imperative that the privacy escalation team resists the urge to restore affected systems and services prematurely. Before initiating the steps to eradicate and remediate the event, the escalation manager and the security engineer must ensure that the procedures have been followed to isolate the affected systems and preserve all forensic information. Once this has been done, compromised systems may be rebuilt and the team should follow remediation procedures.

4. Recovery

   The purpose of the recovery phase is to bring affected systems back into production carefully and ensuring that doing so will not lead to another similar incident. It is essential to test, monitor, and validate that the systems being put back into production will not be compromised by the same or similar IoC (Indicator of Compromise).

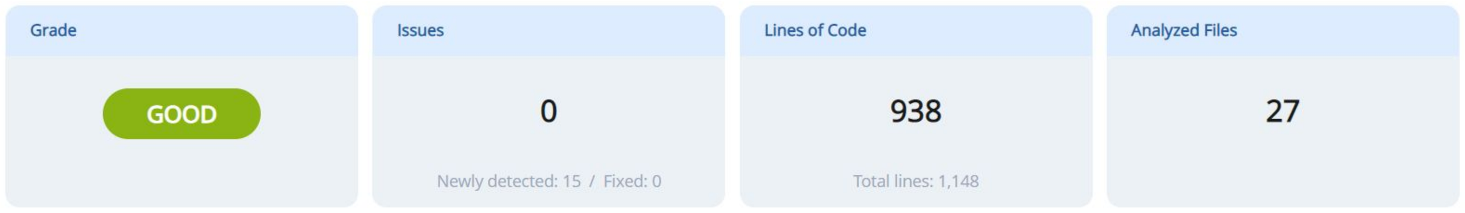
   It is also during this phase that any reports be made and a post-incident follow up be held to discuss the events. During this time, the incident response procedures should be reevaluated and updated as needed.

Final Security Review

Final Dynamic Analysis Review Using Iroh:

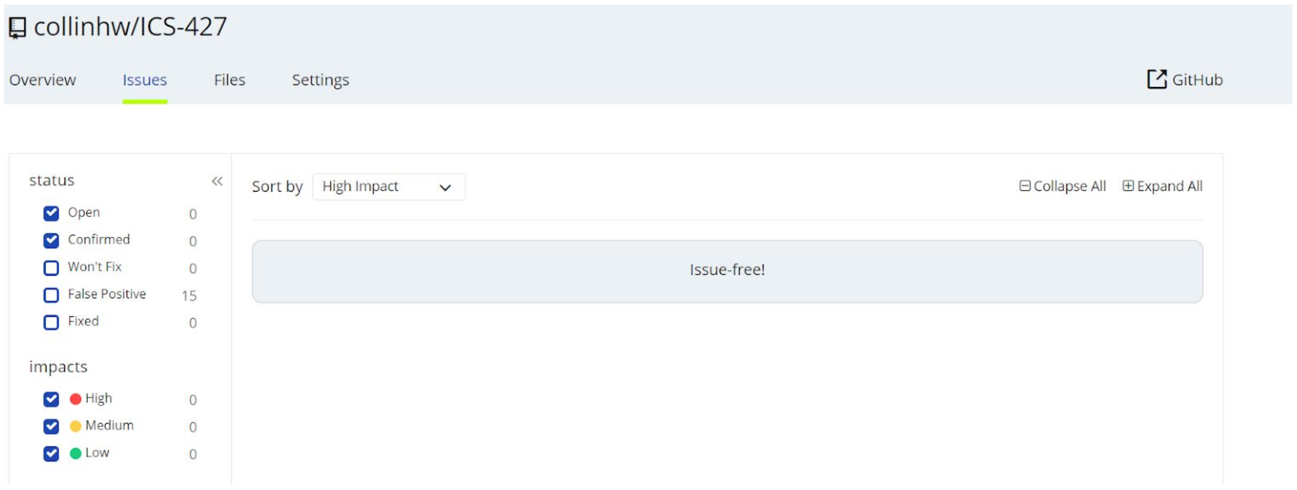
We have continued to use Iroh throughout the second half of the assignment. We were able to identify fallacies in our code and quickly fix them. Overall, as a free open-source tool, Iroh is a great addition to any software developer who wants to learn how to use Dynamic Analysis to improve their work. I would say that there are better tools where Dynamic Analysis is not very necessary in this situation due to the low amount of functions being used. Rather I would say that Static Analysis is a better indicator of how the code is performing rather than looking at Dynamic Analysis.

Final Static Analysis Review Using DeepScan:



Grade	Issues	Lines of Code	Analyzed Files
GOOD	0 Newly detected: 15 / Fixed: 0	938 Total lines: 1,148	27

Going over the previous DeepScan, we have made significant improvements in fixing our issues and developing our fundamentals. We managed to delete any unused declarations and code that would look messy in our final product. After doing another scan we were able to have no new issues. Our DeepScan overall grade changed from Normal → Good with no new issues.


collinhw/ICS-427

Overview Issues Files Settings GitHub

status	
✓ Open	0
✓ Confirmed	0
Won't Fix	0
False Positive	15
Fixed	0

impacts	
✓ ● High	0
✓ ● Medium	0
✓ ● Low	0

Sort by High Impact

Collapse All Expand All

Issue-free!

Threat Modeling:

Going over our original threat modeling diagram, we can identify each threat and dissect how we were able to circumvent these issues. When it comes to data spoofing, repudiation, and tampering, it would be the task of a database administrator to identify if any of the data is faulty or corrupted through SQL injections. To prevent this, we added specific fields where adding certain SQL commands aren't allowed to be put in. For information disclosure, at our app page and Wiki page, we have vowed to be completely transparent with our community. No unneeded data collection shall occur while the end-user is using the application. We respect our user's security and privacy to the fullest. To prevent DDOS attacks, we would have to have our own personal servers and firewalls associated with our network hardware to track the packets that are incoming and limit them through a time-based system by assigning them a specific ID. When going over administrator privileges, there are only two groups, the end-user, and administrators. To prevent users from having admin privileges, we have eliminated direct elevation where users can change groups on the edit page where only an administrator who has been assigned to the administrator group can change permissions.

FSR Grading:

Going over the Final Security Review process, we have determined that we have completed the security milestone that was required during our development. After the milestones were completed we went over our original threat models and were able to ensure that the threats were mitigated. After finishing up our security analysis using DeepScan and Iroh, we were able to clean up security issues prior to our release. Overall, the outcome of this project is a passed FSR as all security measurements and as our own security advisors, we have certified that this project has met all SDL requirements.

Certified Release and Archive Report

The link to the initial release of the program can be found here:
https://github.com/collinhw/ICS-427/releases

Features of the program in the first release (version 1.0) include:

- Ability to sign up for individualized accounts.
- Ability to create a study group for ICS classes (up to ICS 355).
- Ability to view a list of created study groups.

Future development plans for this application include:

- Add more classes to the application (i.e. upper-level ICS, Math, Chemistry, Physics, etc.)
- Ability to edit study groups (in case something changes).
- Ability to create an individualized profile for an account.
    - It is the hope that using the information the user provides (i.e. major, class standing, etc., the application would offer suggestions for study groups that would fit their needs).
- Ability to filter study groups based on class, subject, location, or time.

To install this application, first, install meteor. Next, download the release of the program. In a command line, cd into the app/ directory of the repo and install meteor libraries with the command “meteor npm install”. Finally, once the meteor libraries are installed, start the application using the command “meteor npm run start”.