M.A.C.K.
Memory Leak
Sprint 2 Security Evaluation

Prepared by Andrew Palmer

Capstone Project: Spring 2022

The University of West Florida

Last Modified:

24 March 2022

CIS4595: Capstone Project

Dr. Bernd Owsnicki Klewe

Table of Contents

1. Introduction

All tests were performed on the application that was being hosted by Firebase. All attempted exploits and attacks were performed in such a way that a success would not impact the status of the server or database in any negative or irreversible way. For this evaluation, there were few applicable attacks that I was able to execute due to the application not having all functionalities implemented. In future evaluations, it is likely that more potential exploits and vulnerabilities will be included since there will be more functionality to evaluate.

2. Attempted Exploits/Attacks

a. Cross-site Scripting - *failed*

i. For this, I attempted to inject <script> tags into input fields in an attempt to execute arbitrary JavaScript code through the web application. This did not work, and when performed on the question creation input fields, the resulting question object contained the <script> tags and their content without executing the script within them.

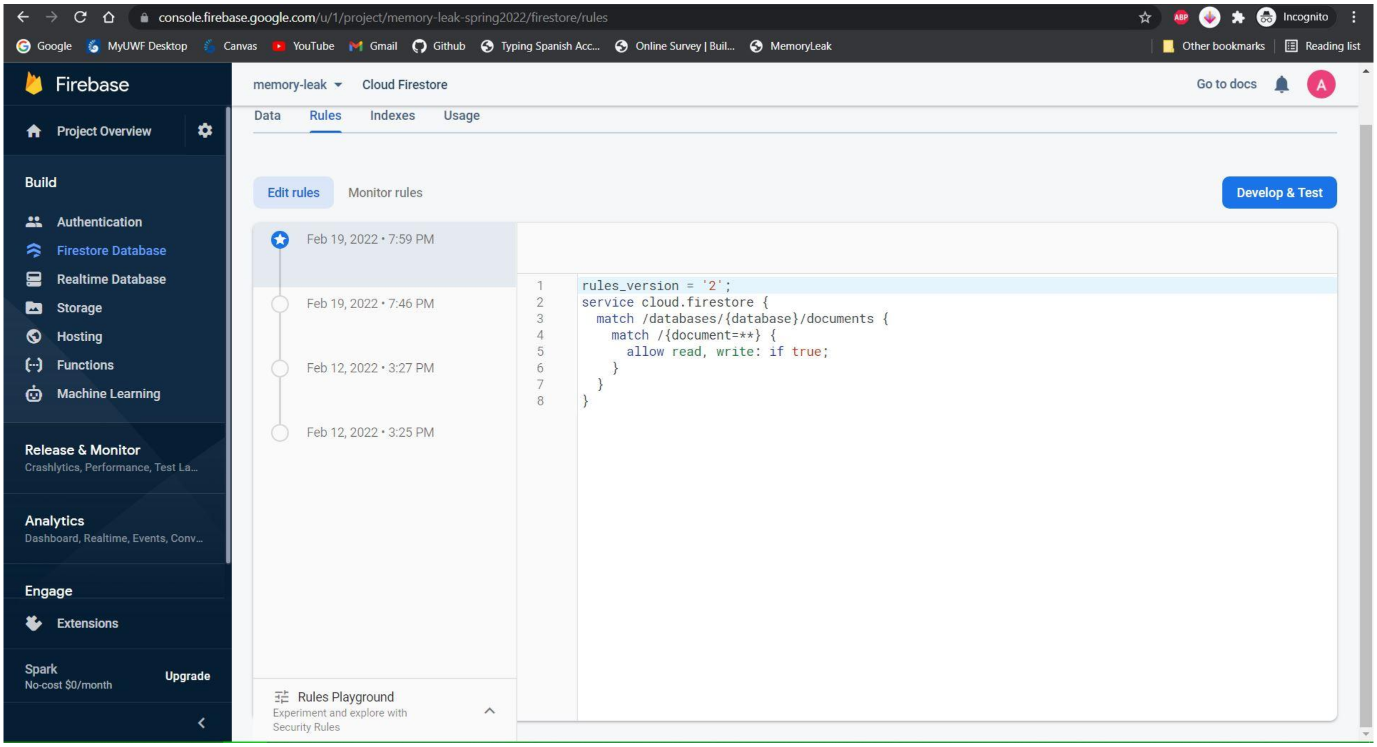
b. DOM-based XSS - *failed*

i. For this attack, I attempted to execute arbitrary JavaScript code by passing it in through the URL. The resulting URL was https://memory-leak-spring2022.web.app/ask#<script>alert("it worked");</script>. This did not cause the application to execute the intended script, and resulted in the ask page rendering as it would without the added <script> tags.

3. Database Vulnerabilities

a. Potential Vulnerability: Firestore Rules

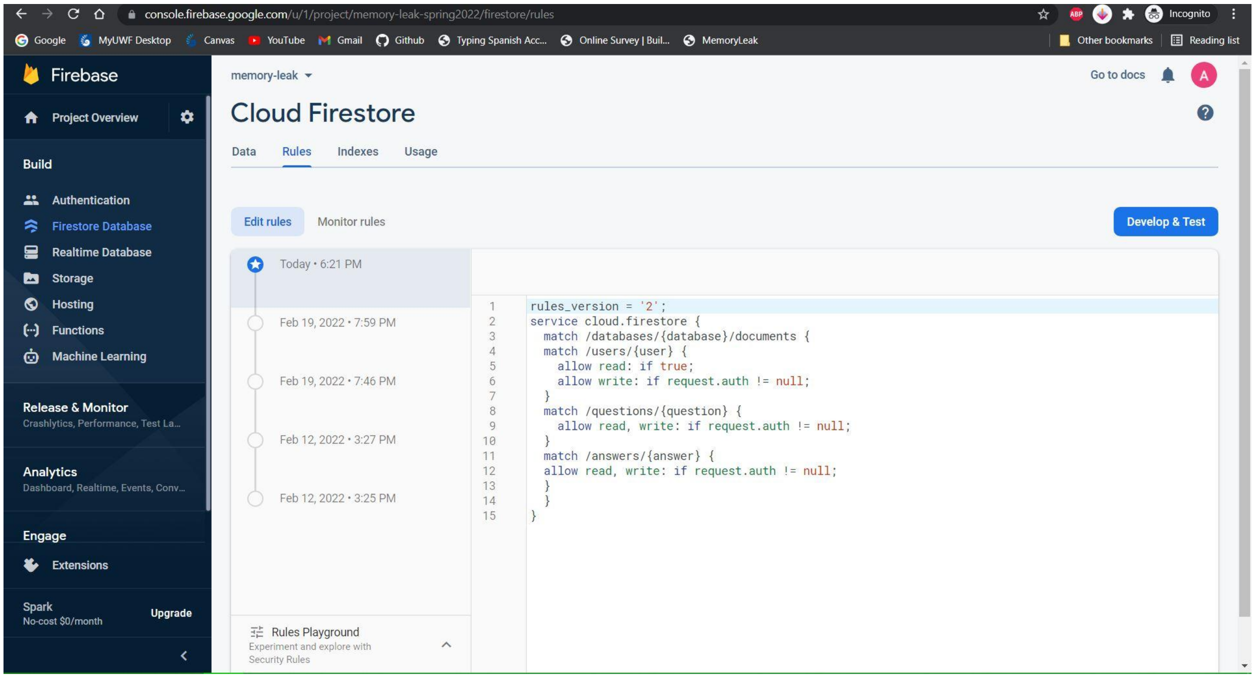
As of the end of Sprint 2, the security rules being implemented for the group's Firestore database allowed for read and write access in all cases. This could allow malicious actors to view and edit the data in our database without authentication, making attribution more difficult in the event of an attack. The figure below illustrates the insecure ruleset that has been in place since February 19, 2022.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

b. Recommended Solution

My recommended security settings to address these vulnerabilities are depicted in the following figure.



```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
  match /users/{user} {
    allow read: if true;
    allow write: if request.auth != null;
  }
  match /questions/{question} {
    allow read, write: if request.auth != null;
  }
  match /answers/{answer} {
  allow read, write: if request.auth != null;
  }
  }
}
```

These rules will only allow reading from the users table when the user is not authenticated. Any other read or write attempts require the user to be authenticated. Currently, our only authentication method is third-party authentication through Google.