Course Matrix/term-group-project-c01w25-project-course-matrix

Security Measures & Testing

1. Authentication, Authorization, and Data Protection Mechanisms

The authentication mechanism used in this project leverages **email and password-based authentication**, managed through Supabase Auth. Supabase provides a comprehensive authentication service that simplifies user management, including account creation, login, and session handling. The key aspects of the authentication process are outlined below:

User Registration (Sign-Up)

- Users provide their **email** and **password** for registration.
- The system invokes **supabase.auth.signUp()** to create a new user account.
- Upon successful registration, a **confirmation email** is sent to the user’s email address. This email contains a link to verify the user's email.
- After clicking the verification link, the user is redirected to a success page.
- A **username** can also be set as additional user metadata during registration.

User Authentication (Login)

- Users log in by providing their **email** and **password.**
- The **supabase.auth.signInWithPassword()** method is used to authenticate the user.
- On successful authentication, an **access token** and **refresh token** are generated.
- The **refresh token** is stored securely in an **HTTP-only, secure cookie** to maintain the user’s session, preventing access from JavaScript and mitigating XSS (cross-site scripting) attacks.

3. Email Verification

- The email verification step adds an extra layer of security to the authentication process.

- Users must click the link sent to their email to activate their account.

4. Session Management

- Once the user is logged in, their session is maintained using **JWT (JSON Web Tokens)**. The **access token** is used to authenticate the user for subsequent requests, and the **refresh token** is used to refresh the session when needed.

2. Security Practices Followed

1. **HTTPS**:
    - The application is configured to use **HTTPS** in production environments (`secure: process.env.NODE_ENV === 'production'`), ensuring that all communication between the client and the server is encrypted. This prevents man-in-the-middle (MITM) attacks and protects user credentials during transmission.
2. **Password Hashing**:
    - Supabase handles password hashing automatically as part of the authentication process. This means that **user passwords** are never stored in plaintext but are hashed and stored securely in the database, preventing the exposure of sensitive information.
3. **Email Verification**:
    - **Email verification** is used as an additional layer of security. After registering, users must verify their email address before they can access the system. This ensures that only valid email addresses are used to create accounts and prevents the use of fake or mistyped emails.
4. **HTTP-Only Cookies**:
    - The **refresh token** is stored in **HTTP-only, secure cookies**, which prevents it from being accessed by JavaScript. This mitigates the risk of **Cross-Site Scripting (XSS)** attacks where an attacker might attempt to steal tokens via malicious scripts.
5. **SameSite Cookies**:
    - The **`sameSite: 'strict'`** cookie setting is used, which prevents the refresh token cookie from being sent along with cross-site

requests. This helps to prevent **Cross-Site Request Forgery (CSRF)** attacks.

6. **Session Management:**
   - The application utilizes **JWT-based (JSON Web Tokens)** authentication, which ensures that users remain logged in across requests without needing to constantly re-enter credentials. The refresh token is securely stored and used to maintain the session, reducing the likelihood of session hijacking.
7. **Input Validation:**
   - Input validation is performed both client-side and server-side to prevent the submission of malicious or invalid data. For example, missing or invalid fields in the registration or login forms are rejected, which helps prevent attacks like **SQL Injection.**
8. **Error Handling:**
   - Comprehensive error handling is implemented to ensure that sensitive information (such as database or server errors) is not exposed to the user. Generic error messages are returned to the client to avoid leaking any internal details that could be exploited by attackers.
9. **Secure Password Management:**
   - While Supabase automatically handles password management and security (including salting and hashing), it is important to note that the system adheres to best practices for password security, ensuring passwords are never stored or transmitted in an insecure manner.

3. Basic Security Tests

SQL Injection

Since we are using Supabase as our database management system, we benefit from built-in protections against SQL Injection attacks. Nevertheless, we selected a subset of endpoints to test using **sqlmap**, an open-source tool specifically designed to detect SQL Injection vulnerabilities.

Test 1 (User Login):

sqlrequest.txt:

```
POST http://localhost:8081/auth/login HTTP/1.1

host: localhost:8081

{"email":"thomas.yang@mail.utoronto.ca","password":"Password123!"}
```

Terminal Command:

```
python sqlmap.py -r sqlrequest.txt --level=5 --risk=3
--proxy=http://127.0.0.1:8080 --dbms=PostgreSQL --ignore-code=401
```

Key Options & Their Purpose

- **-r sqlrequest.txt** → Reads HTTP request from **sqlrequest.txt** instead of specifying a URL.
- **--level=5** → Uses the **most extensive** SQL injection tests (1 = basic, 5 = maximum).
- **--risk=3** → Enables **high-risk** SQL injection techniques (1 = low, 3 = high).
- **--proxy=http://127.0.0.1:8080** → Routes requests through external proxy.
- **--dbms=PostgreSQL** → Targets **only PostgreSQL databases**
- **--ignore-code=401** → Ignore **HTTP 401 (Unauthorized)** errors since that is the return code when login has invalid credentials.

After over 2000 tests against SQL Injection, here is the result from the terminal:

```
[CRITICAL] all tested parameters do not appear to be injectable.
```

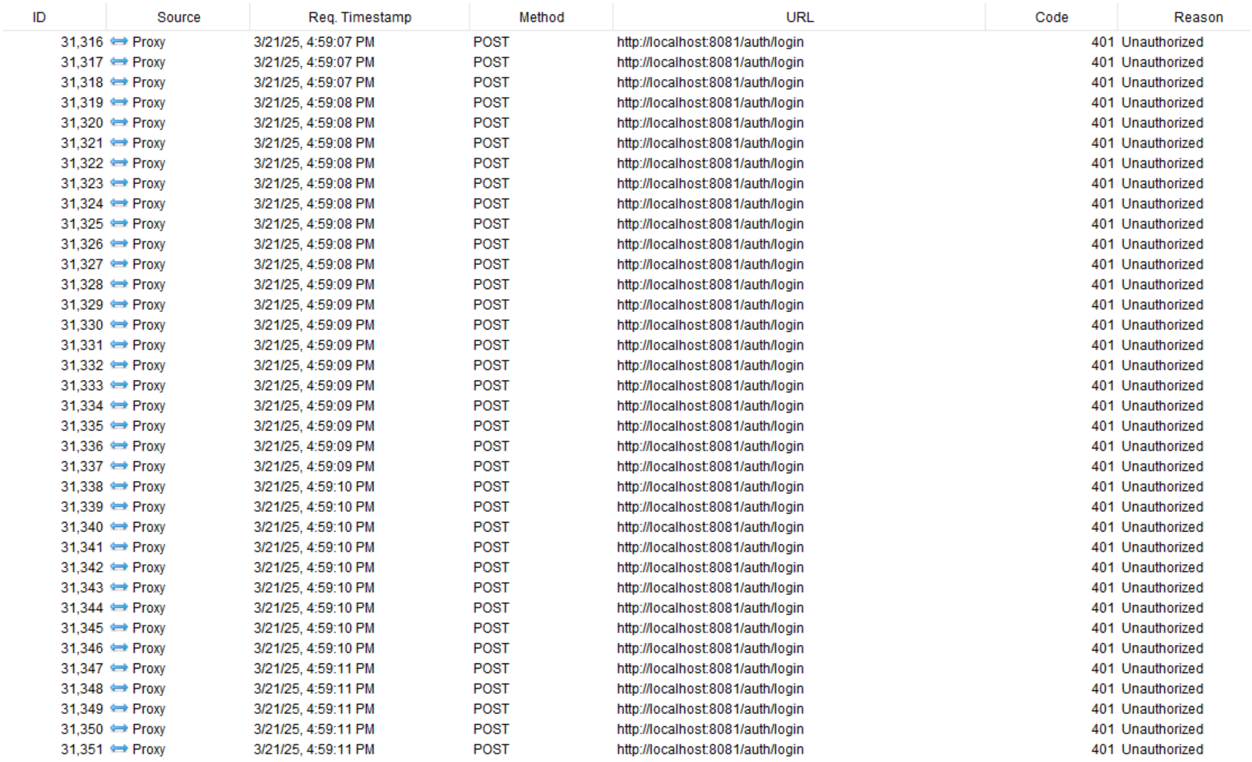
Test 2 (Get events from user calendar):

Terminal Command:

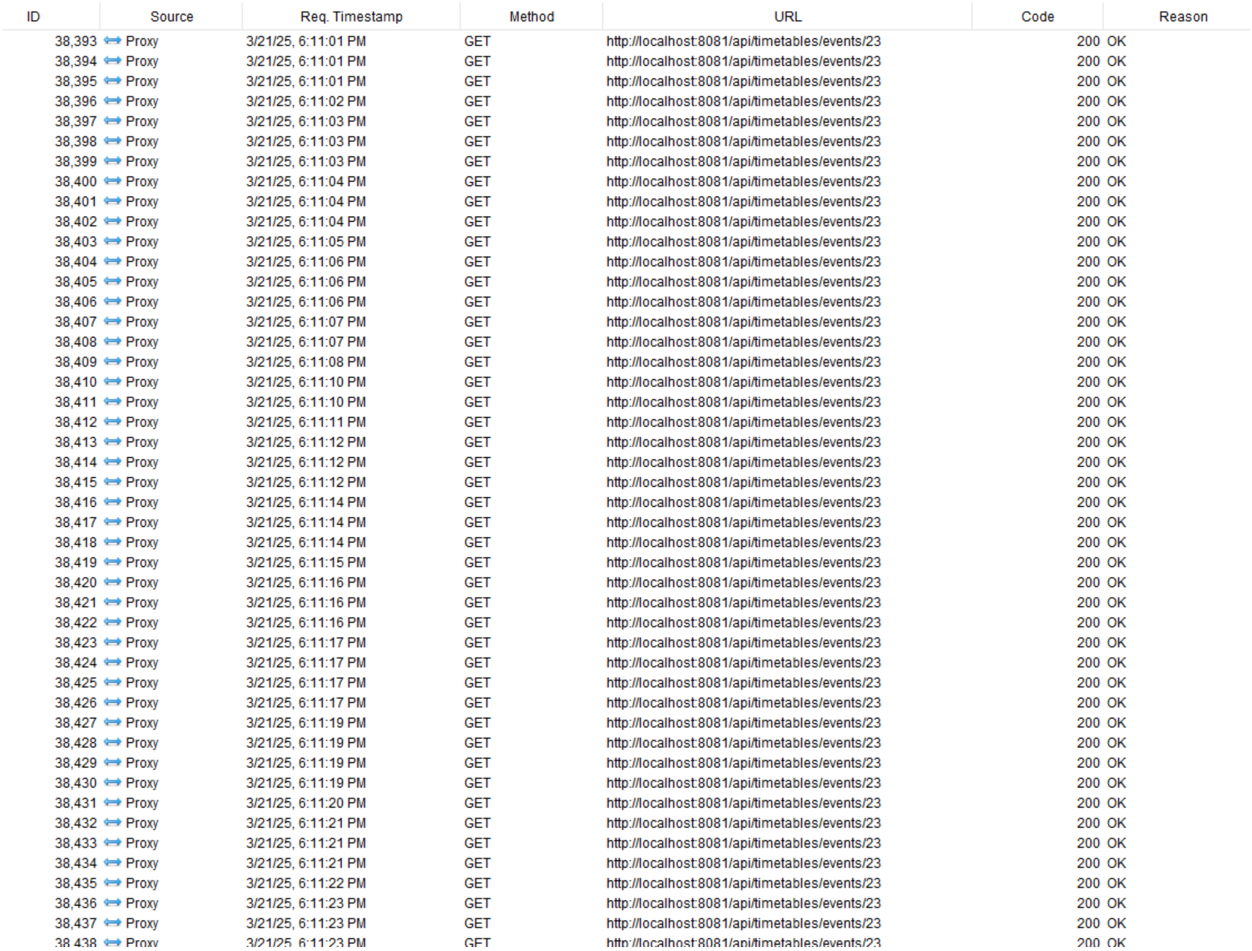
```
python sqlmap.py -u
"http://localhost:8081/api/timetables/events/23" --level=5 --risk=3
--proxy=http://127.0.0.1:8080 --dbms=PostgreSQL --ignore-code=401
--cookie="refresh_token=[token]"
```

[CRITICAL] all tested parameters do not appear to be injectable.

Screenshots below:



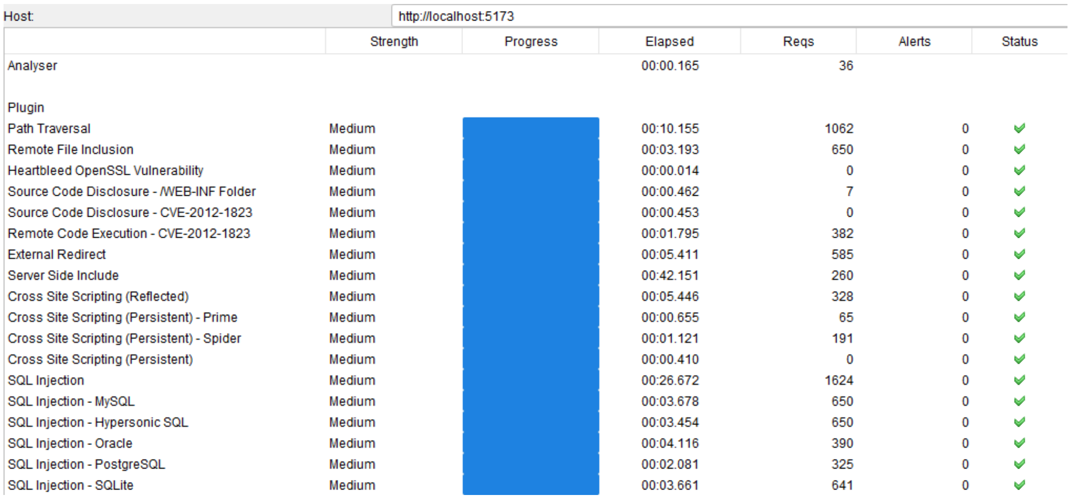
ID	Source	Req. Timestamp	Method	URL	Code	Reason
31,316	⟷ Proxy	3/21/25, 4:59:07 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,317	⟷ Proxy	3/21/25, 4:59:07 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,318	⟷ Proxy	3/21/25, 4:59:07 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,319	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,320	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,321	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,322	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,323	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,324	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,325	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,326	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,327	⟷ Proxy	3/21/25, 4:59:08 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,328	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,329	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,330	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,331	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,332	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,333	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,334	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,335	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,336	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,337	⟷ Proxy	3/21/25, 4:59:09 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,338	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,339	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,340	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,341	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,342	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,343	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,344	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,345	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,346	⟷ Proxy	3/21/25, 4:59:10 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,347	⟷ Proxy	3/21/25, 4:59:11 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,348	⟷ Proxy	3/21/25, 4:59:11 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,349	⟷ Proxy	3/21/25, 4:59:11 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,350	⟷ Proxy	3/21/25, 4:59:11 PM	POST	http://localhost:8081/auth/login	401	Unauthorized
31,351	⟷ Proxy	3/21/25, 4:59:11 PM	POST	http://localhost:8081/auth/login	401	Unauthorized



ID	Source	Req. Timestamp	Method	URL	Code	Reason
38,393	⇄ Proxy	3/21/25, 6:11:01 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,394	⇄ Proxy	3/21/25, 6:11:01 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,395	⇄ Proxy	3/21/25, 6:11:01 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,396	⇄ Proxy	3/21/25, 6:11:02 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,397	⇄ Proxy	3/21/25, 6:11:03 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,398	⇄ Proxy	3/21/25, 6:11:03 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,399	⇄ Proxy	3/21/25, 6:11:03 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,400	⇄ Proxy	3/21/25, 6:11:04 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,401	⇄ Proxy	3/21/25, 6:11:04 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,402	⇄ Proxy	3/21/25, 6:11:04 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,403	⇄ Proxy	3/21/25, 6:11:05 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,404	⇄ Proxy	3/21/25, 6:11:06 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,405	⇄ Proxy	3/21/25, 6:11:06 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,406	⇄ Proxy	3/21/25, 6:11:06 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,407	⇄ Proxy	3/21/25, 6:11:07 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,408	⇄ Proxy	3/21/25, 6:11:07 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,409	⇄ Proxy	3/21/25, 6:11:08 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,410	⇄ Proxy	3/21/25, 6:11:10 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,411	⇄ Proxy	3/21/25, 6:11:10 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,412	⇄ Proxy	3/21/25, 6:11:11 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,413	⇄ Proxy	3/21/25, 6:11:12 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,414	⇄ Proxy	3/21/25, 6:11:12 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,415	⇄ Proxy	3/21/25, 6:11:12 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,416	⇄ Proxy	3/21/25, 6:11:14 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,417	⇄ Proxy	3/21/25, 6:11:14 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,418	⇄ Proxy	3/21/25, 6:11:14 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,419	⇄ Proxy	3/21/25, 6:11:15 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,420	⇄ Proxy	3/21/25, 6:11:16 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,421	⇄ Proxy	3/21/25, 6:11:16 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,422	⇄ Proxy	3/21/25, 6:11:16 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,423	⇄ Proxy	3/21/25, 6:11:17 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,424	⇄ Proxy	3/21/25, 6:11:17 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,425	⇄ Proxy	3/21/25, 6:11:17 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,426	⇄ Proxy	3/21/25, 6:11:17 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,427	⇄ Proxy	3/21/25, 6:11:19 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,428	⇄ Proxy	3/21/25, 6:11:19 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,429	⇄ Proxy	3/21/25, 6:11:19 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,430	⇄ Proxy	3/21/25, 6:11:19 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,431	⇄ Proxy	3/21/25, 6:11:20 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,432	⇄ Proxy	3/21/25, 6:11:21 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,433	⇄ Proxy	3/21/25, 6:11:21 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,434	⇄ Proxy	3/21/25, 6:11:21 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,435	⇄ Proxy	3/21/25, 6:11:22 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,436	⇄ Proxy	3/21/25, 6:11:23 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,437	⇄ Proxy	3/21/25, 6:11:23 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK
38,438	⇄ Proxy	3/21/25, 6:11:23 PM	GET	http://localhost:8081/api/timetables/events/23	200	OK

General Testing through ZAP:

We also performed generic security tests using ZAP, and the results were satisfactory. Below is a screenshot of the test results:



Host: http://localhost:5173

	Strength	Progress	Elapsed	Reqs	Alerts	Status
Analyser			00:00.165	36		
Plugin						
Path Traversal	Medium		00:10.155	1062	0	✓
Remote File Inclusion	Medium		00:03.193	650	0	✓
Heartbleed OpenSSL Vulnerability	Medium		00:00.014	0	0	✓
Source Code Disclosure - /WEB-INF Folder	Medium		00:00.462	7	0	✓
Source Code Disclosure - CVE-2012-1823	Medium		00:00.453	0	0	✓
Remote Code Execution - CVE-2012-1823	Medium		00:01.795	382	0	✓
External Redirect	Medium		00:05.411	585	0	✓
Server Side Include	Medium		00:42.151	260	0	✓
Cross Site Scripting (Reflected)	Medium		00:05.446	328	0	✓
Cross Site Scripting (Persistent) - Prime	Medium		00:00.655	65	0	✓
Cross Site Scripting (Persistent) - Spider	Medium		00:01.121	191	0	✓
Cross Site Scripting (Persistent)	Medium		00:00.410	0	0	✓
SQL Injection	Medium		00:26.672	1624	0	✓
SQL Injection - MySQL	Medium		00:03.678	650	0	✓
SQL Injection - Hypersonic SQL	Medium		00:03.454	650	0	✓
SQL Injection - Oracle	Medium		00:04.116	390	0	✓
SQL Injection - PostgreSQL	Medium		00:02.081	325	0	✓
SQL Injection - SQLite	Medium		00:03.661	641	0	✓