SecureLayer7

Time and Again, Securing you

Kimai Time Tracking Web Application

Vulnerability Assessment and Penetration Testing Report

Table of Contents

Executive Summary

SecureLayer7 is pleased to present the results of the Penetration Testing Assessment conducted on the Kimai Time Tracking application and service from 1st January 2023 to 31st January 2023. The assessment was conducted by a team of 1 Penetration Tester, 1 Team Lead, and 1 Project Manager. The assessment was performed remotely, and the methodology and approach used for the assessment was White Box Penetration Testing.

The assessment began with the Penetration Testers and Team Lead working together to identify potential vulnerabilities in the application and service. This was done by utilizing a combination of automated tools and manual testing methods. Once potential vulnerabilities were identified, they were analyzed and evaluated by the Penetration Testers and Team Lead to determine the impact and likelihood of exploitation.

The Project Manager was responsible for managing the overall project plan, monitoring, and controlling project documentation, and providing high-quality deliverables. The Project Manager also worked closely with the Kimai Time Tracking team to ensure that the assessment was conducted in a manner that was both thorough and non-disruptive to their daily operations.

The results of the assessment were compiled into a comprehensive report that includes an executive summary, detailed descriptions of the vulnerabilities identified, and recommendations for remediation. The report also includes detailed information on the methodology and approach used for the assessment, as well as the qualifications and experience of the SecureLayer7 team members who conducted the assessment.

It is important to note that the findings in this report reflect the conditions found during the assessment and do not necessarily reflect current conditions. The security test results and findings provided in this report are valid for the period during which the assessment was carried out and are based on the information provided for the assessment.

SecureLayer7 would like to thank the Kimai Time Tracking team for their cooperation and support during the assessment. We are confident that the information provided in this report will be valuable in helping the Kimai Time Tracking team to improve the security posture of their application and service. We stand ready to assist in any way possible as the Kimai Time Tracking team works to address the issues identified during the assessment.

Scope

Scope Details
White-Box Tests against Kimai Time Tracking https://demo-plugins.kimai.org/en/login

Penetration Test Details:

Activity Date(s)	1st January 2023 – 31st January 2023

Coverage

White-Box Tests against Kimai Time Tracking

- The web application is examined for Unauthenticated Access to API endpoints from the Custom Fields module by requesting to fetch Custom Field details without the cookie header. After initiating the request without the cookie, the application responded with a code with an error message as “Access denied” in the response headers.
- The application was tested for data-storage injection, such as SQL injection on search functionality in various modules. It was observed that the web application does not display any SQL-based errors after injecting the payload into numerous fields. Even after injecting the time-based SQL injection payload, the web application does not show any delay in response. The web application also targets using time-based and union-based payloads, but the application has protection from SQL injection. The web application did not show any response latency even after inserting the time-based SQL injection payload.
- The application is tested for Stored Cross-Site Scripting vulnerability in the Custom Fields module by introducing the XSS payload into a number of updates and adding Custom Fields request form fields. The application has implemented server-side validation on user input, which prevents requests from being processed if they contain any special characters. Additionally, it was noted that the application uses HTML encoding to reflect user input and has added the necessary security headers to protect against Cross-site Scripting (XSS) attacks.
- The application was tested for session timeout issues. After successfully logging into the application, if the user does not perform any activity on the application, then the application terminates the active session.
- The application was tested for The JS files used in the application are analyzed for sensitive information disclosure such as API keys, Hardcoded Credentials, Session Tokens, etc. But no sensitive information is leaked through the JS files.
- The application was tested for Vulnerable Outdated Components by analyzing the client-side comments and using automated extensions. It was observed that the application did not use any outdated JS libraries, and there were no vulnerable components.
- The application was tested for Improper Error Handling vulnerability on various endpoints by testing the application's error-handling behavior with invalid user input via HTTP request modification. Later verify how the application reacts to those user inputs and whether it can get internal server information via a verbose error message. It was observed that the application returns only generic error messages without sensitive information.
- The application was tested for server-side template injection with various payloads to check if the payload gets executed or gets any type of server template debugs error message. Used a blind template payload to check if any interaction happened on the collaborator server but noticed that the template injection payload didn't get executed from the application.
- The application was also tested for unauthenticated access control issues by initiating the requests without a cookies header. After initiating a request without the cookie, the application responds with a 403 status code and an error message as “Access denied” in response headers.
- The application was tested for path traversal vulnerability via URL parameter. Tested for path traversal vulnerability to read internal files but noticed that the application returns a 400-error message while tested for path traversal. Further, when trying to encode the path traversal payload with URL encodes, but unable to read local files from the server.
- The application was tested for Code Injection attacks by providing the code injection payloads at various injection endpoints. It was observed that the application does not accept the value as payload and returns a forbidden error. The application has proper server-side validation for user input and privilege, which does not allow the user to execute code and read any arbitrary internal files from the server.
- The Administration module from the application was tested for unauthenticated access to the files. It was discovered that the administration module could export data to an Excel or XML file. By removing the cookies header from the request, when trying to access the file, it was observed that the application only allows authenticated users to access the file; it shows a 501, not implemented error.
- The Administration module from the application was tested for XML Entities injection attack. It was observed that the application has the functionality to upload the XML file. By changing the file content,

SecureLayer7 tried injecting malicious content to extract the sensitive file. The application checks the file content and shows the error "error importing data."

- The Administration module from the application was tested for Client-side validation bypass. It was observed that the application validates the special characters to avoid Cross-Site scripting and CSV injection attacks. SecureLayer7 attempted to bypass the validation by intercepting the request. It was observed that the application has validated the input and shows the error "Invalid characters Error."
- The Administration module of the application was tested for Time-based SQL injection and error Blind SQL injection. While testing, SecureLayer7 observed that the application has the functionality to search the data. By injection of the time-based SQL injection payload, it was observed that the application shows the error "Invalid keyword search configuration."
- The Kimai application is tested for SQL injection vulnerabilities by manually reviewing source code using Visual Studio Code IDE. Further analysis revealed that the application uses parametrized SQL statements.
- The Kimai application is tested for server-side template injection vulnerabilities by manually reviewing source code using Visual Studio Code IDE. Further analysis did not reveal any user-controllable input that renders directly in the template engine.
- The Kimai application's calendar description and my task description is tested for stored cross-site scripting vulnerability by manually entering different JavaScript payloads and found that application does not execute Script payloads.
- The Kimai application's invoice and calendar description module are tested for HTML injection to local file inclusion vulnerability by manually entering different file paths in iFrame source payloads during further exploitation and found that the application does not render local file contains.
- The application was tested for SQL injection vulnerability on various parameters of the search module and other API endpoints by injecting a time-based SQL injection payload and observing the response time. After injecting the time-based SQL injection payload, there was no delay observed in response time. Hence the application is not vulnerable to SQL injection.
- The application was tested for Cross-Site Request Forgery (CSRF) on the "Timesheet Edit" functionality by tampering with the Anti-CSRF token implementation and trying to bypass the same by removing the token value, changing the request method and sending the null token in the request. But for all the test cases, the application responded with "The CSRF token is invalid". Hence, the application is not to CSRF.
- The application was tested for Cross-Site Scripting (XSS) vulnerability on the "Timesheet Create" functionality by introducing XSS payloads in all the form fields. It was observed that the application has appropriately sanitized the user input and implemented server-side validation. Hence the application is not vulnerable to XSS.
- The application was tested for Insecure Direct Object Reference (IDOR) vulnerability on the "Profile Edit" functionality by changing the username on the URL to that of a different user. But the application responded with 403 forbidden. Hence the application is secured against IDOR.

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g., *KT-01-001*) for the purpose of facilitating any future follow-up correspondence.

1. KT-01-001: Stored HTML Injection on Description Parameter at Calendar Edit Record Module (*Info*)

The Calendar module's description input field in the web application lacked input sanitization checks. This vulnerability was discovered during further analysis and it was found that attackers could inject HTML code into the user-controllable input fields, leading to permanent embedding of the code into the web page. As a result, raw HTML syntax is parsed every time other users interact with the affected page.

To demonstrate this vulnerability, the application was tested for HTML Injection by updating the details of an existing task with an HTML payload of <h1>HTML</h1> in the description parameter. The result showed that the application successfully rendered the HTML payload in the calendar module.

Affected Module:

- *https://demo-plugins.kimai.org/en/calendar/*

Affected File:

The HTML Injection vulnerability exists in the calendar module, and below is one sample affected source code file demonstrating the missing sanitization on the user-controlled input field.

- *kimai/assets/js/widgets/KimaiCalendar.js/*

Affected Code:

```
          <li>` + this.options['translations']['project'] + `: `
          +escaper.escapeForHtml(eventObj.project) + `</li>
          <li>` + this.options['translations']['activity'] + `: ` +
          escaper.escapeForHtml(eventObj.activity) + `</li>
          </ul>` +
          [...]
          (eventObj.description !== null || eventObj.tags.length > 0 ? '<hr>':
'')+
          (eventObj.description ? '<p>' + eventObj.description + '</p>': '') +
          (eventObj.tags !== null && eventObj.tags.length > 0 ? '
          <span class=" badge bg-green">' + eventObj.tags.join('</span>
          <span class=" badge bg-green">') + '</span>': '') + `
          </div>`;
          [...]
```

Mitigation:

The mitigation for the vulnerability of the Calendar module's description input field in the web application involves two main strategies. The first strategy is filter input on arrival. This means that when the application receives user input, it should filter it strictly based on what is considered a valid input. This can prevent attackers from injecting malicious HTML code into the input fields, ensuring the security of the application.

The second strategy is output encoding. This involves encoding user-controllable data when it is output on an HTML page. This will prevent the data from being interpreted as active content and thus prevent any malicious code from executing on the affected page. Encoding the output will ensure that the raw HTML syntax is not parsed whenever other users interact with the affected page, thus increasing the security of the application.

Conclusion

The web application was thoroughly tested for various security vulnerabilities, including unauthenticated access control, data-storage injection, Cross-Site Scripting, session timeout, sensitive information disclosure, vulnerable outdated components, improper error handling, server-side template injection, path traversal, code injection, and administration module access control.

The testing process involved both automated and manual methods, and the results showed that the application has implemented proper security measures to prevent various attacks. For example, the application returns a 403 status code and error message "Access Denied" for unauthenticated access control requests, and it has proper server-side validation for user input to prevent SQL injection, Cross-Site Scripting, and code injection attacks.

Moreover, the Kimai application, which is part of the web application, was also tested for SQL injection and server-side template injection vulnerabilities. The manual source code review showed that the application uses parametrized SQL statements and does not have any user-controllable input that renders directly in the template engine. The calendar description and invoice description were also tested for stored Cross-Site Scripting vulnerability and found to be secure.

In conclusion, the web application has robust security measures in place and has successfully passed multiple security vulnerability tests. The results show that the application is secure against various security threats and can safely be used.