Lab 4

Cybersecurity Testing Part II

Group 1

, Abdul-Ramon Adeyemi Folarin, &

, , &

School of Information Technology Administration & Security, Seneca Polytechnic

WAS705NBB: Application Security Methodology

Professors Suranjit Paul & Mark Shtern

July 1, 2026

Student Integrity Statement: I/we declare that the attached assignment is my/our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) or distributed to other students

Task Checklist

Table 1 Lab 4 Task Completion Checklist

#	Task	Status
1	Authentication testing: default and known credentials (WSTG 4.4.2)	**Done**
2	Authentication testing: weak password policy (WSTG 4.4.7)	**Done**
3	Authentication testing: session timeout (WSTG 4.6.7)	**Done**
4	Authorization testing: vertical bypass (WSTG 4.5.2)	**Done**
5	Authorization testing: horizontal bypass (WSTG 4.5.2)	**Done**
6	Authorization testing: access to administrative functions (WSTG 4.5.2)	**Done**
7	Authorization testing: special request header handling (WSTG 4.5.2)	**Done**
8	Authorization testing: resources assigned to a different role (WSTG 4.5.2)	**Done**
9	Authorization testing: privilege escalation (WSTG 4.5.3)	**Done**
10	Secret management: storage of application secrets (WSTG 4.2.2)	**Done**
11	Secret management: debug and development artifacts (WSTG 4.2.2)	**Done**
12	Secret management: secrets in container logs (WSTG 4.2.2)	**Done**
13	Secret management: API token storage review	**Done**
14	Secret management: encryption key (APP_SECRET) management	**Done**
15	Secret management: weak encryption and password hashing (WSTG 4.9.4)	**Done**
16	Secret management: source code review for weak algorithms (WSTG 4.9.4)	**Done**
17	API security: IDOR/BOLA modify another user's object (WSTG 4.5.4)	**Done**
18	API security: IDOR/BOLA delete another user's object (WSTG 4.5.4)	**Done**
19	API security: authentication rate limiting (WSTG 4.4.3)	**Done**
20	API security: error handling (WSTG 4.8.1)	**Done**
21	API security: input validation (via error-handling fuzzing)	**Done**
22	API security: SQL injection, manual and sqlmap (WSTG 4.7.5)	**Done**
23	Fuzzing: API parameter (timesheet ID)	**Done**
24	Fuzzing: token field (bearer token)	**Done**
25	Fuzzing: authentication endpoint (login form)	**Done**
26	Fuzzing: secret-related input	**Done**
27	Static code analysis: Snyk dependency scan	**Done**
28	Static code analysis: Semgrep source scan	**Done**
29	Static code analysis: ESLint security scan	**Done**
30	Static code analysis: Bandit applicability review	**Done**
31	Security architecture: authentication/authorization flow diagram	**Done**
32	Security architecture: security architecture diagram	**Done**

#	Task	Status
33	Security unit testing and automation proposal	**Done**
34	Threat coverage: STRIDE mapping and justification (Lab 2)	**Done**
35	Vulnerability coverage: known-vulnerability mapping (Lab 1)	**Done**
36	Gap analysis and additional controls	**Done**
37	MFA / 2FA testing	**Done**

Executive Summary

This report documents the Lab 4 cybersecurity assessment of Kimai 2.59.0, an open-source time-tracking application developed in PHP on the Symfony framework and deployed via Docker Compose. Testing was conducted by Group 1 of WAS705 (Application Security Methodology, Summer 2026) at Seneca Polytechnic, under the supervision of Professors Suranjit Paul and Mark Shtern. The assessment targeted authentication and authorization mechanisms, secret management, API security, fuzzing, and static code analysis, in accordance with OWASP Web Security Testing Guide (WSTG) v4.2 and the OWASP Top 10 2025.

Testing was performed from a Kali Linux host at 192.168.43.145 against the Kimai Docker deployment on Ubuntu at 192.168.43.146 (host port 8920). The assessment used Burp Suite, curl, ffuf, sqlmap, Snyk, Semgrep, and ESLint with the security plugin. All testing was conducted in an authorized laboratory environment.

No authentication bypass, SQL injection, or privilege escalation was identified in the tested scenarios. Server-side role-based access control (RBAC) consistently returned HTTP 403 for unauthorized requests across both the web interface and the REST API. Object-level authorization was enforced on all tested CRUD operations, and the Symfony debug profiler was confirmed disabled in production. Passwords are hashed with bcrypt at cost factor 13.

The assessment identified three categories of weakness. The most significant is the storage of API tokens in plaintext in the database: while database access is a prerequisite, a single database disclosure yields directly usable credentials, unlike bcrypt-hashed user passwords. The tested deployment also transmits traffic over plain HTTP without TLS, and the Docker entrypoint logs the administrative password in cleartext on every startup. These are deployment-level hardening gaps rather than Kimai application code flaws. A dependency scan by Snyk identified a Critical deserialization vulnerability in the bundled phpoffice/phpspreadsheet library; exploitability within Kimai was not independently confirmed, but the dependency should be upgraded. Password policy enforcement is limited to a minimum length of 8 characters with no complexity requirement, and no API rate limiting was observed during a bounded 30-request test.

The table below summarizes all findings. Recommended controls are described in detail in the Gap Analysis section.

Table 2 Findings Summary

Finding	Severity	OWASP 2025	Recommended Control
Plaintext API token storage at rest	**Medium**	A04 Cryptographic Failures	Hash tokens using SHA-256; compare hashes on auth
Plain HTTP; no TLS on tested deployment	**Medium**	A02 Security Misconfiguration	Deploy behind HTTPS reverse proxy; set Secure flag
Admin password in plaintext env vars and startup logs	**Low**	A02 Security Misconfiguration	Use secrets manager; suppress password from logs
Database credentials in plaintext env vars	**Low**	A02 Security Misconfiguration	Use Docker secrets or external vault
Length-only password policy (8 chars, no complexity)	**Low**	A07 Authentication Failures	Increase minimum length; block common passwords; enforce MFA
7-day PHP session GC lifetime; exact idle timeout unverified	**Low**	A07 Authentication Failures	Configure explicit shorter idle timeout; verify rejection
No API rate limiting observed (bounded 30-req test)	**Low**	A07 Authentication Failures	Apply per-token/per-IP rate limiting on API endpoints
Snyk: Critical CVE in phpoffice/phpspreadsheet (deser.)	**Undetermined**	A03 Software Supply Chain	Upgrade phpoffice/phpspreadsheet to patched release
Snyk: Medium CVE in mpdf/mpdf (remote file inclusion)	**Undetermined**	A03 Software Supply Chain	Upgrade mpdf/mpdf to patched release
Snyk: Medium CVEs in dompurify JavaScript (XSS)	**Undetermined**	A03 Software Supply Chain	Upgrade dompurify to patched release
Semgrep: 40 Doctrine query-builder warnings (likely FP)	**Informational**	A05 Injection	Manual review of flagged QueryBuilder calls
ESLint: 45 object-injection / unsafe-regex warnings (likely FP)	**Informational**	A05 Injection	Trace bracket-notation access to confirm/refute

Table of Contents

Task Checklist 2
Executive Summary 4
Table Of Figures 8
Table Of Tables 11
1. Authentication Testing 12
1.1 Default and Known Credentials (WSTG-ATHN-02) - OWASP A02:2025 Security Misconfiguration 12
1.2 Weak Password Policy (WSTG-ATHN-07) - OWASP A07:2025 Authentication Failures 14
1.3 Session Timeout (WSTG-SESS-07) - OWASP A07:2025 Authentication Failures 16
2.1 Vertical Access Control Bypass (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control 21
2.2 Horizontal Access Control Bypass (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control 22
2.3 Special Request Header Handling (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control 23
2.4 Resources Assigned to a Different Role (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control 24
2.5 Privilege Escalation (WSTG-ATHZ-03) - OWASP A01:2025 Broken Access Control 24
3. Secret Management Testing 26
3.1 Storage of Application Secrets (WSTG-CONF-02) - OWASP A02:2025 Security Misconfiguration 26
3.2 Debug and Development Artifacts (WSTG-CONF-02) - OWASP A02:2025 Security Misconfiguration 26
3.3 Secrets in Container Logs (WSTG-CONF-02) - OWASP A02:2025 / A09:2025 27
3.4 API Token Storage (WSTG-CONF-02) - OWASP A04:2025 Cryptographic Failures 28
3.5 Encryption Key (APP_SECRET) Management (WSTG-CONF-02) - OWASP A04:2025 Cryptographic Failures 29
3.6 Weak Encryption and Password Hashing (WSTG-CRYP-04) - OWASP A04:2025 Cryptographic Failures 29
3.7 Source Code Review for Weak Algorithms (WSTG-CRYP-04) - OWASP A04:2025 Cryptographic Failures 29
4.1 Insecure Direct Object References / BOLA (WSTG 4.5.4) - OWASP A01:2025 Broken Access Control 31
4.2 API Authentication Rate Limiting (WSTG-ATHN-03) - OWASP A07:2025 Authentication Failures 32
4.3 Error Handling and Malformed Input (WSTG-ERRH-01) - OWASP A10:2025 34

4.4 SQL Injection Testing (WSTG-INPV-05) - OWASP A05:2025 Injection 35
5. Fuzzing..... 38
5.1 API Parameter Fuzzing (Timesheet ID) - OWASP A01:2025 Broken Access Control..... 38
5.2 Authentication Endpoint Fuzzing (Login Form) - OWASP A07:2025 Authentication Failures..... 39
5.3 Secret-Related Input Fuzzing - OWASP A07:2025 Authentication Failures..... 40
6. Static Code Analysis..... 41
6.1 Software Composition Analysis with Snyk - OWASP A03:2025 Software Supply Chain 41
6.2 Static Analysis with Semgrep - OWASP A05:2025 Injection..... 42
6.3 ESLint Security Plugin - OWASP A05:2025 Injection..... 43
6.4 Tool Applicability: Bandit (Not Applicable)..... 44
7.1 Authentication and Authorization Flow Diagram..... 45
7.2 Security Architecture Diagram 46
8. Security Unit Testing and Automation Proposal 47
8.1 Input Validation Tests..... 47
8.2 Token Verification Tests..... 47
8.3 Role-Based Access Enforcement Tests..... 47
9. Threat and Vulnerability Coverage..... 48
9.1 STRIDE Threat Coverage (from Lab 2) 48
9.2 Known Vulnerability Coverage (from Lab 1)..... 50
10. Gap Analysis and Additional Controls 51
10.1 Plaintext API Token Storage at Rest (Medium - A04:2025 Cryptographic Failures).. 51
10.2 Plain HTTP; No TLS (Medium - A02:2025 Security Misconfiguration)..... 51
10.3 Cleartext Secrets in Environment Variables and Logs (Low - A02:2025 Security Misconfiguration)..... 51
10.4 Length-Only Password Policy (Low - A07:2025 Authentication Failures) 51
10.5 No Observed API Rate Limiting (Low - A07:2025 Authentication Failures)..... 52
10.6 Session Garbage-Collection Lifetime of 7 Days (Low - A07:2025 Authentication Failures) 52
10.7 Outdated Dependencies with Known CVEs (Undetermined - A03:2025 Software Supply Chain) 52
Appendix A: Generative AI Usage Disclosure..... 53
Appendix B: References..... 54

Table Of Figures

Figure 1 Kimai login page at localhost:8920/en/login with the known administrative username admin@kimai.local entered over plain HTTP. 12
Figure 2 Authenticated Kimai dashboard showing the AD account with Administration and System menu items visible, confirming successful login. 13
Figure 3 Deployment configuration file (docker-compose.yml) showing ADMINPASS and DATABASE_URL stored in plaintext. 14
Figure 4 15
Figure 5 Kimai accepting the common weak password 'password' on the alice account, confirmed by the browser password-manager save prompt 16
Figure 6 Server-side session configuration showing PHP session.gc_maxlifetime set to 604800 seconds (7 days) and session.cookie_lifetime of 0. 17
Figure 7 Burp Repeater showing the captured KIMAI_SESSION cookie accepted before session invalidation, returning HTTP 302 redirect to the authenticated timesheet page. 18
Figure 8 Database query confirming the active session record exists in kimai2_sessions before invalidation 19
Figure 9 Database query confirming the active session row was deleted from kimai2_sessions, returning a count of 0. 19
Figure 10 Burp Repeater showing the same KIMAI_SESSION cookie rejected after the server-side session row was deleted, returning HTTP 302 redirect to /en/login. 20
Figure 11 Admin-only Users management page at /en/admin/user/ accessed while logged in as admin, listing all three accounts 21
Figure 12 Standard user alice receiving HTTP 403 'Page is restricted' when requesting the admin-only Users URL directly. 22
Figure 13 Standard user alice's API token receiving HTTP 403 Forbidden when requesting bob's timesheet entry via GET /api/timesheets/2. 22
Figure 14 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to create a new super-admin user via POST /api/users. 23
Figure 15 Identical HTTP 403 response when alice's denied request is replayed with spoofed override and proxy headers (X-Original-URL, X-Rewrite-URL, X-Forwarded-For, X-Remote-IP, X-Client-IP). 23
Figure 16 Standard user alice's API token receiving HTTP 200 OK with a read-only customer list from GET /api/customers, confirming expected read visibility 24
Figure 17 The same alice token receiving HTTP 403 Forbidden when attempting to create a customer via POST /api/customers, confirming write operations remain restricted. 24
Figure 18 Standard user alice attempting to grant herself ROLE_SUPER_ADMIN via PATCH /api/users/2, rejected with HTTP 400 because the roles field is not part of her editable form. 25
Figure 19 Kimai container environment variables showing ADMINPASS and DATABASE_URL stored in plaintext, with APP_ENV correctly set to prod 26
Figure 20 Symfony debug profiler request returning HTTP 404 Not Found, confirming the development profiler is disabled in production 27
Figure 21 Container startup logs repeatedly printing the admin user-creation command with the plaintext password; APP_SECRET shown only as 'auto-generated' without its value. 28
Figure 22 The kimai2_access_token table showing API tokens stored in plaintext, with the lab4 token matching the value used for API authentication. 28

Figure 23 APP_SECRET stored in /opt/kimai/var/data/.appsecret with owner-only permissions (-rw-------) and present in the production cache. 29
Figure 24 Stored password hash for the alice account showing bcrypt with cost factor 13 ($2y$13$), confirming a strong adaptive hashing algorithm. 29
Figure 25 Recursive source search of Kimai's PHP src/ directory for weak cryptographic algorithm keywords returning no matches. 30
Figure 26 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to modify bob's timesheet entry via PATCH /api/timesheets/2. 31
Figure 27 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to delete bob's timesheet entry via DELETE /api/timesheets/2. 31
Figure 28 Thirty repeated authenticated requests to /api/users/me using Alice's valid token returning HTTP 200 with no throttling observed. 32
Figure 29 Thirty repeated invalid API-token requests returning HTTP 401 without throttling or delay increase. 33
Figure 30 Baseline PATCH request showing alice successfully updating her own timesheet entry via /api/timesheets/3 with HTTP 200 OK. 34
Figure 31 Kimai returning HTTP 400 Bad Request with structured validation error 'Please enter a valid date and time' when the begin field is set to 'not-a-date'. 34
Figure 32 Follow-up GET request confirming the malformed PATCH did not change alice's timesheet; original values remain. 35
Figure 33 Container log review showing the malformed PATCH returned HTTP 400 without a related exception or stack trace. 35
Figure 34 Single-quote injection in the term parameter of GET /api/timesheets returning HTTP 200 with an empty result set and no SQL error. 35
Figure 35 sqlmap testing the term parameter at level 2 / risk 2 against the Kimai API, concluding the parameter does not appear to be injectable. 36
Figure 36 ffuf fuzzing the timesheet ID parameter from 1 to 50 using alice's token, returning HTTP 200 only for alice's entries (IDs 3 and 4) and HTTP 403 for other users' entries. 38
Figure 37 ffuf fuzzing the bearer token field with ten guessed and random values, all returning HTTP 401 Unauthorized. 39
Figure 38 Login page response showing 'Invalid credentials' after submitting alice's username with an incorrect password, confirming the request reached authentication logic. 39
Figure 39 Successful login response confirming valid credentials redirect to the application root and issue authenticated session cookies. 40
Figure 40 ffuf fuzzing the API bearer token with 999 values from the SecLists darkweb2017 leaked-credentials wordlist, with zero HTTP 200 responses. 40
Figure 41 Snyk dependency scan of Kimai 2.59.0 source reporting a Critical deserialization vulnerability in phpoffice/phpspreadsheet and a Medium remote-file-inclusion in mpdf/mpdf, along with Medium JavaScript issues. 41
Figure 42 Semgrep scan of Kimai's PHP source (src/) running 85 rules across 1041 files, reporting 40 findings predominantly consisting of Doctrine QueryBuilder string-concatenation warnings. 42
Figure 43 ESLint security-plugin scan of Kimai's JavaScript (assets/js/) reporting 45 warnings and 0 errors, with no eval or child-process patterns detected. 43

Figure 44 Authentication and authorization flow for Kimai, showing the web (session-based) and API (token-based) paths, the server-side RBAC layer, and the effective controls and identified weaknesses on each path..... 45
Figure 45 Security architecture of the Kimai deployment showing the tester environment, the two-container Docker network, data flows, trust boundaries, and the location of effective controls and identified weaknesses..... 46

Table Of Tables

Table 1 Lab 4 Task Completion Checklist..... 2
Table 2 Findings Summary..... 5
Table 3 STRIDE Threat Coverage..... 48
Table 4 Known Vulnerability Coverage..... 50

1. Authentication Testing

1.1 Default and Known Credentials (WSTG-ATHN-02) - OWASP A02:2025 Security Misconfiguration

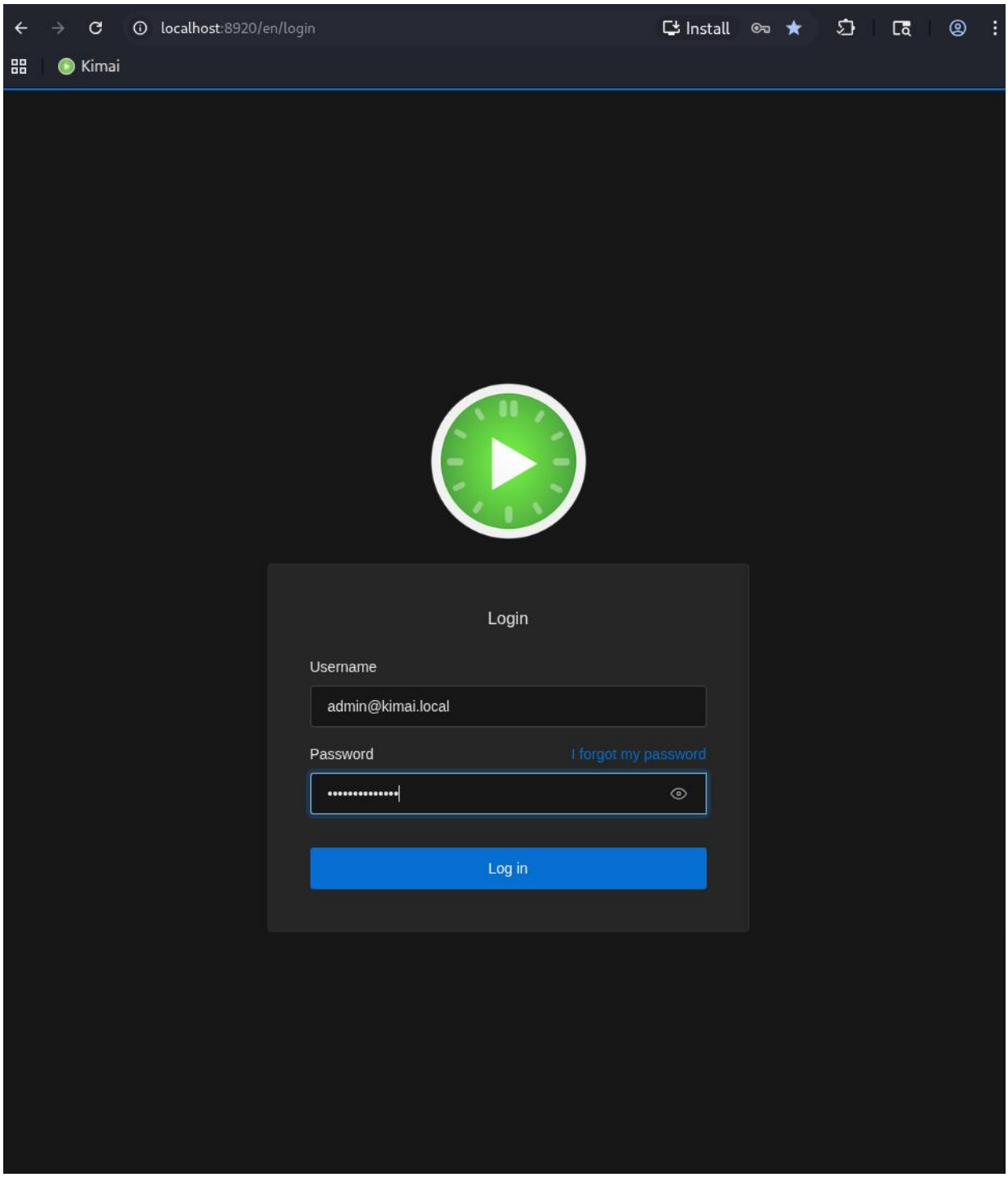


Figure 1 Kimai login page at localhost:8920/en/login with the known administrative username admin@kimai.local entered over plain HTTP.

Figure 1 shows that the testing began by attempting to authenticate to the Kimai administrative interface using the credentials defined in the application's deployment configuration. The username admin@kimai.local was entered on the login form, which is served over plain HTTP rather than HTTPS. These credentials were not guessed; they are the values set by the deployment and never changed after installation, which is the exact condition WSTG 4.4.2 identifies as the underlying weakness.

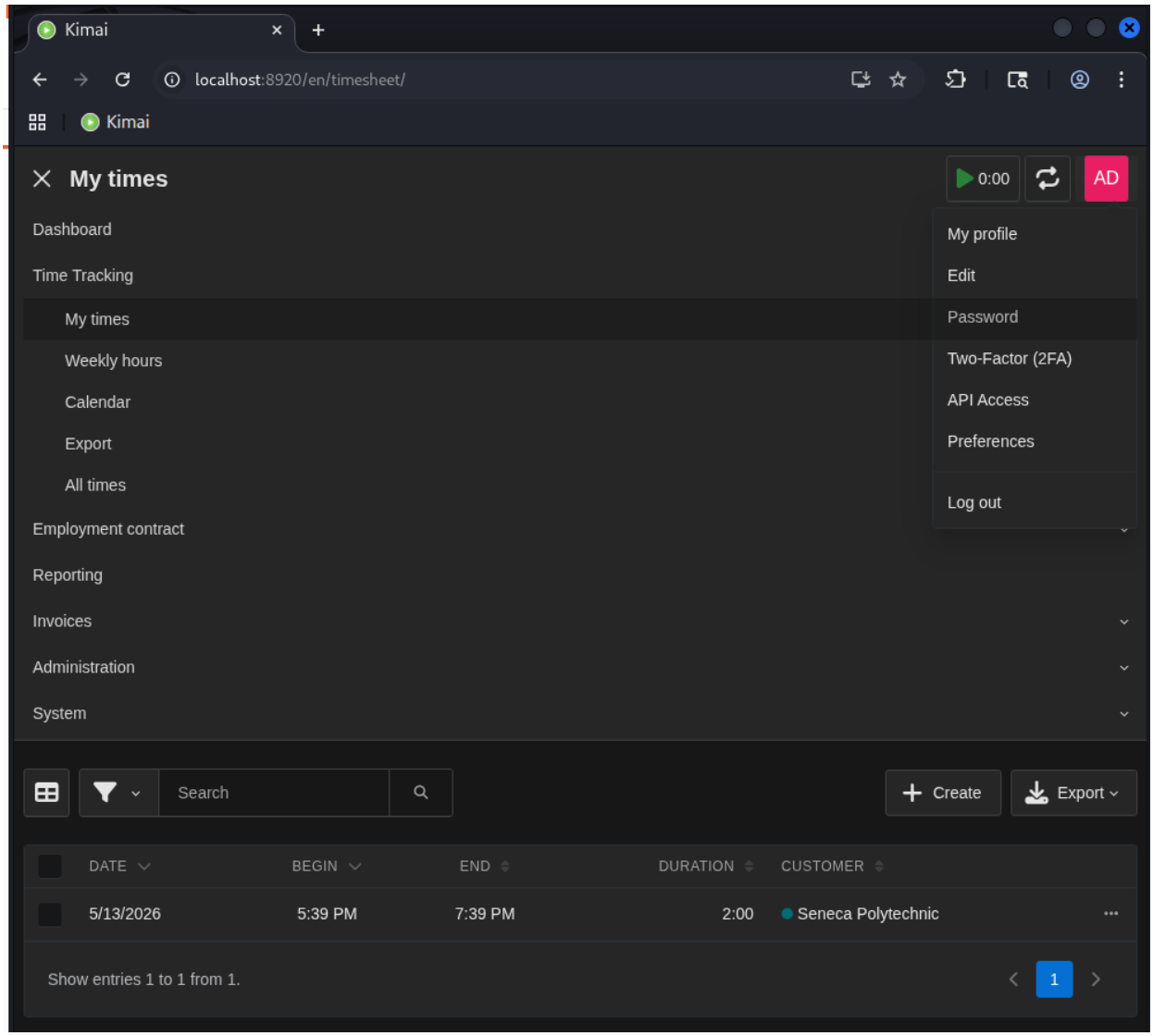


Figure 2 Authenticated Kimai dashboard showing the AD account with Administration and System menu items visible, confirming successful login.

Figure 2 shows the dashboard after login shows the account signed in as AD, with Administration and System entries visible in the sidebar. These menu items are only shown to accounts with administrative privileges, confirming the login succeeded with an administrative-level account.



Figure 3 Deployment configuration file (docker-compose.yml) showing ADMINPASS and DATABASE_URL stored in plaintext.

Figure 3 shows the deployment file defines the administrative password directly as ADMINPASS: changemeplease, and the database credentials appear inline in the DATABASE_URL value (user kimai, password kimai). The password used in Figure 1 matches this hardcoded value exactly, confirming the administrative account was provisioned with a known password and never required to change it. This is a deployment hardening gap rather than a Kimai code flaw. Real-world severity is Low, mapping to OWASP A02:2025 Security Misconfiguration.

1.2 Weak Password Policy (WSTG-ATHN-07) - OWASP A07:2025 Authentication Failures

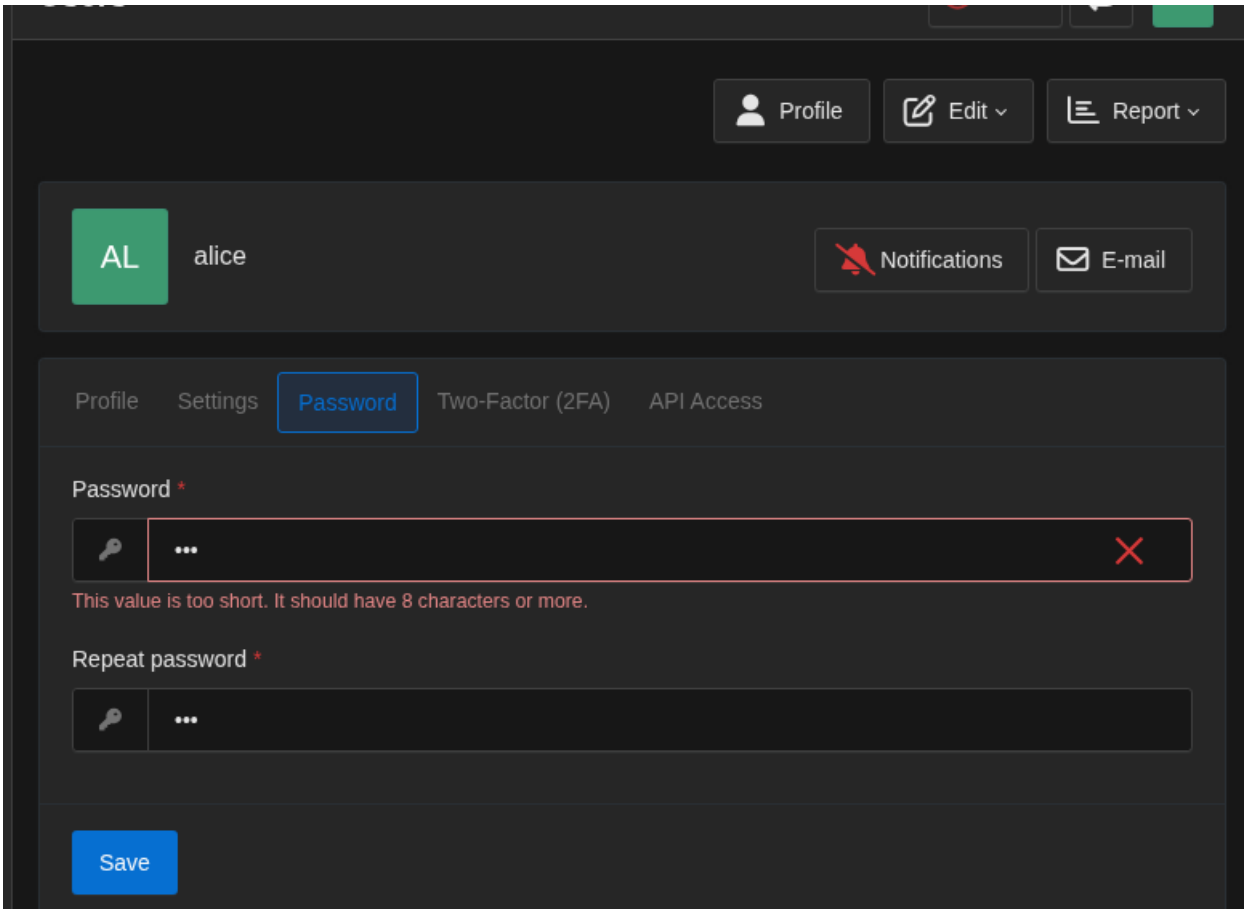


Figure 4

Figure 4. Kimai rejecting a 3-character password on the alice account with the validation message 'This value is too short. It should have 8 characters or more.'

Figure 4 shows WSTG 4.4.7 assesses whether the application enforces a password policy covering length and complexity. A 3-character password was submitted on the alice account password-change page. Kimai rejected it with the message shown in Figure 4, confirming a minimum length of 8 characters is enforced. This is positive evidence of a length-based control, severity Informational.

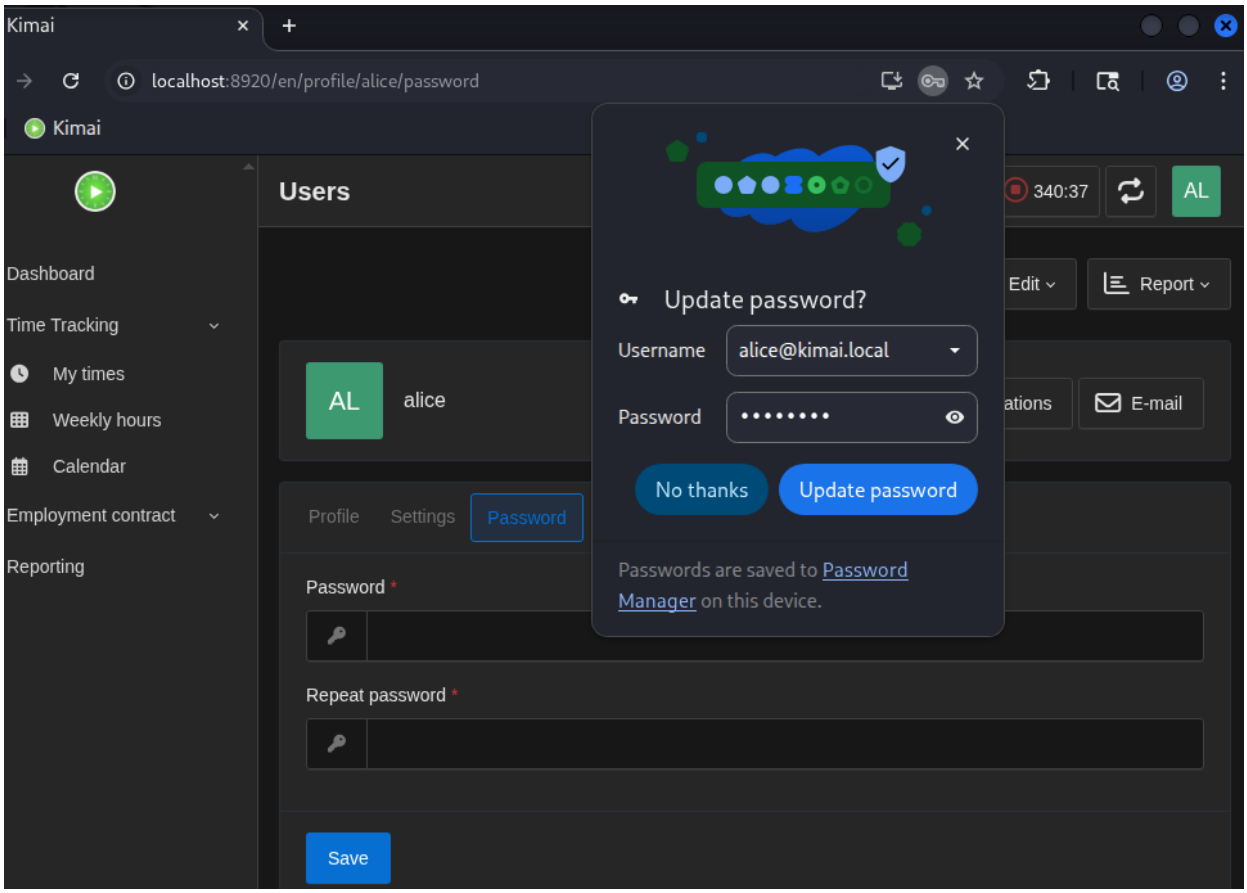


Figure 5 Kimai accepting the common weak password 'password' on the alice account, confirmed by the browser password-manager save prompt.

Figure 5 shows to assess the strength of the policy beyond length, the well-known weak password 'password' (8 characters) was entered. Kimai accepted it, indicated by the form submitting successfully and the browser offering to save the credential. This shows the policy enforces a minimum length but does not enforce complexity requirements or block common weak passwords. Severity Low, mapping to OWASP A07:2025 Authentication Failures. The gap analysis recommends a longer minimum length, passphrase support, blocking common passwords, and MFA for sensitive accounts.

1.3 Session Timeout (WSTG-SESS-07) - OWASP A07:2025 Authentication Failures

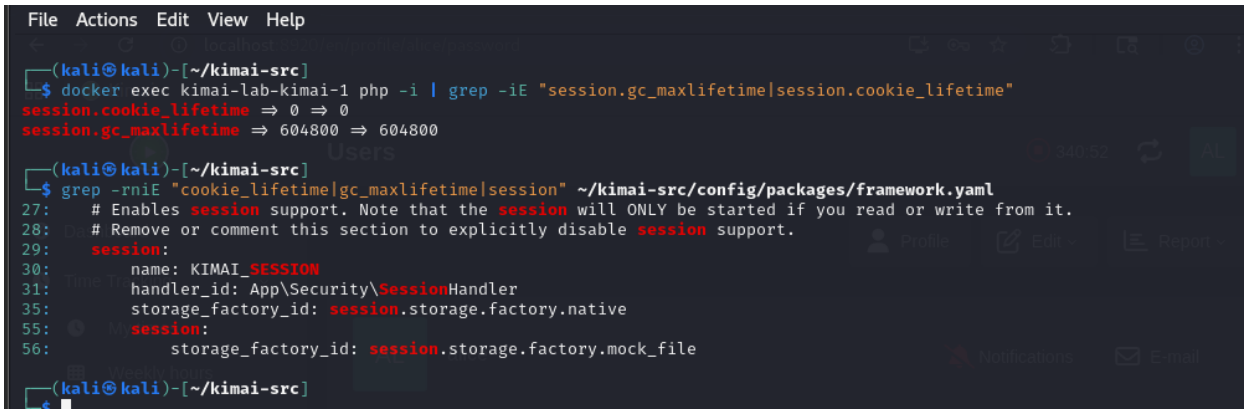


Figure 6 Server-side session configuration showing PHP session.gc_maxlifetime set to 604800 seconds (7 days) and session.cookie_lifetime of 0.

Figure 6 shows running Kimai container showed session.gc_maxlifetime set to 604800 seconds, equal to 7 days. The session.cookie_lifetime value of 0 means the session cookie is non-persistent and is removed when the browser closes. The Symfony framework.yaml file showed no shorter explicit session lifetime override.

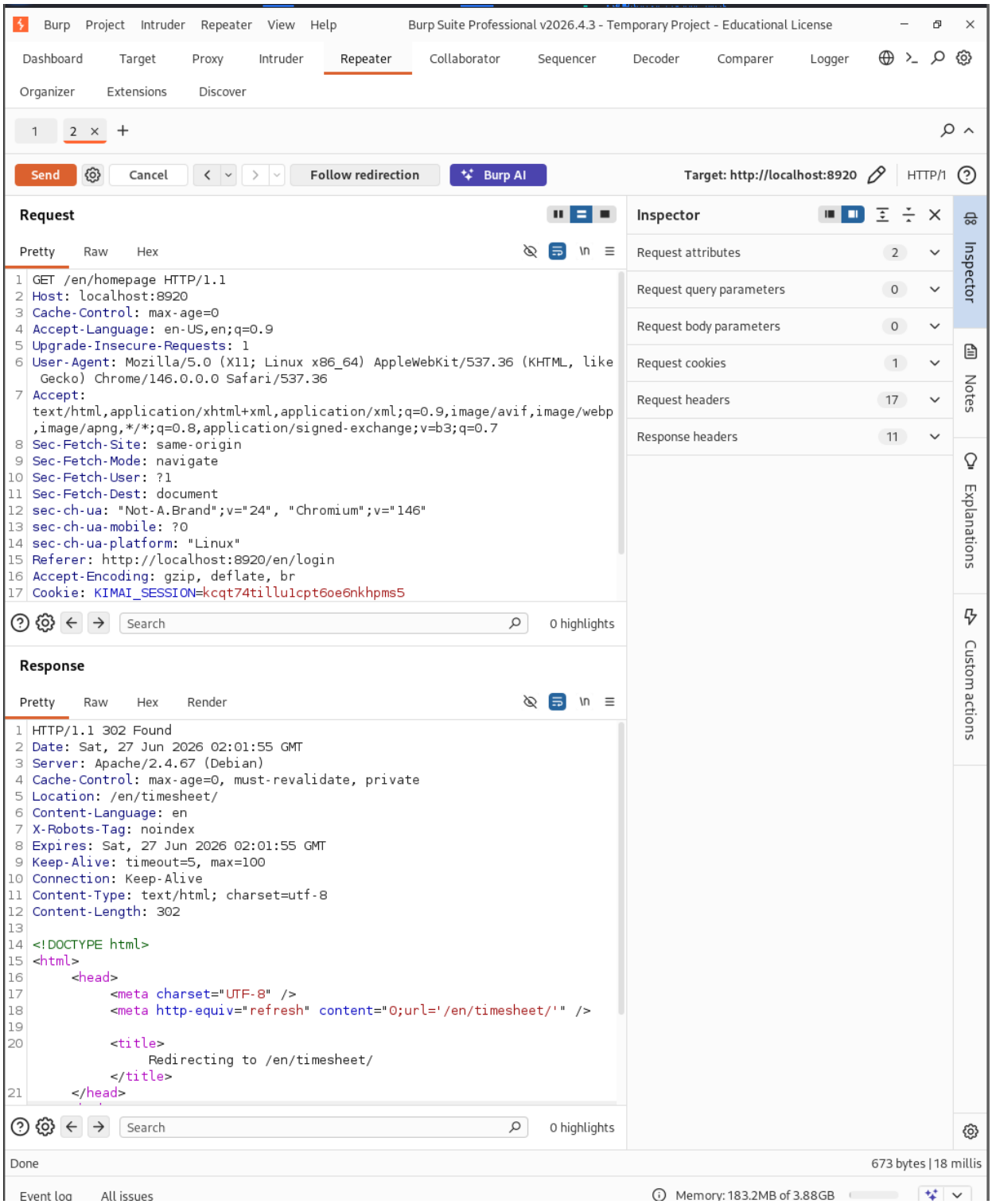


Figure 7 Burp Repeater showing the captured KIMAI_SESSION cookie accepted before session invalidation, returning HTTP 302 redirect to the authenticated timesheet page.

Figure 7 shows logged-in Kimai request was captured and replayed in Burp Repeater using the KIMAI_SESSION cookie. The request was sent to /en/homepage, and Kimai returned HTTP/1.1

302 Found with a redirect to /en/timesheet/. This shows that the captured session cookie was valid and accepted by the server before the session invalidation test was performed.



Figure 8 Database query confirming the active session record exists in kimai2_sessions before invalidation

Figure 8 shows kimai2_sessions table was queried for the active KIMAI_SESSION value used in Burp Repeater. The matching row was found with a last_seen timestamp, lifetime value, and stored session data length. This confirms that the session accepted by Burp Repeater was backed by an active server-side session record before the invalidation test.



Figure 9 Database query confirming the active session row was deleted from kimai2_sessions, returning a count of 0.

Figure 9 shows the matching row for the active KIMAI_SESSION value was deleted from the kimai2_sessions table. A follow-up count query returned 0, confirming that the server-side session record no longer existed before the same Burp Repeater request was replayed.

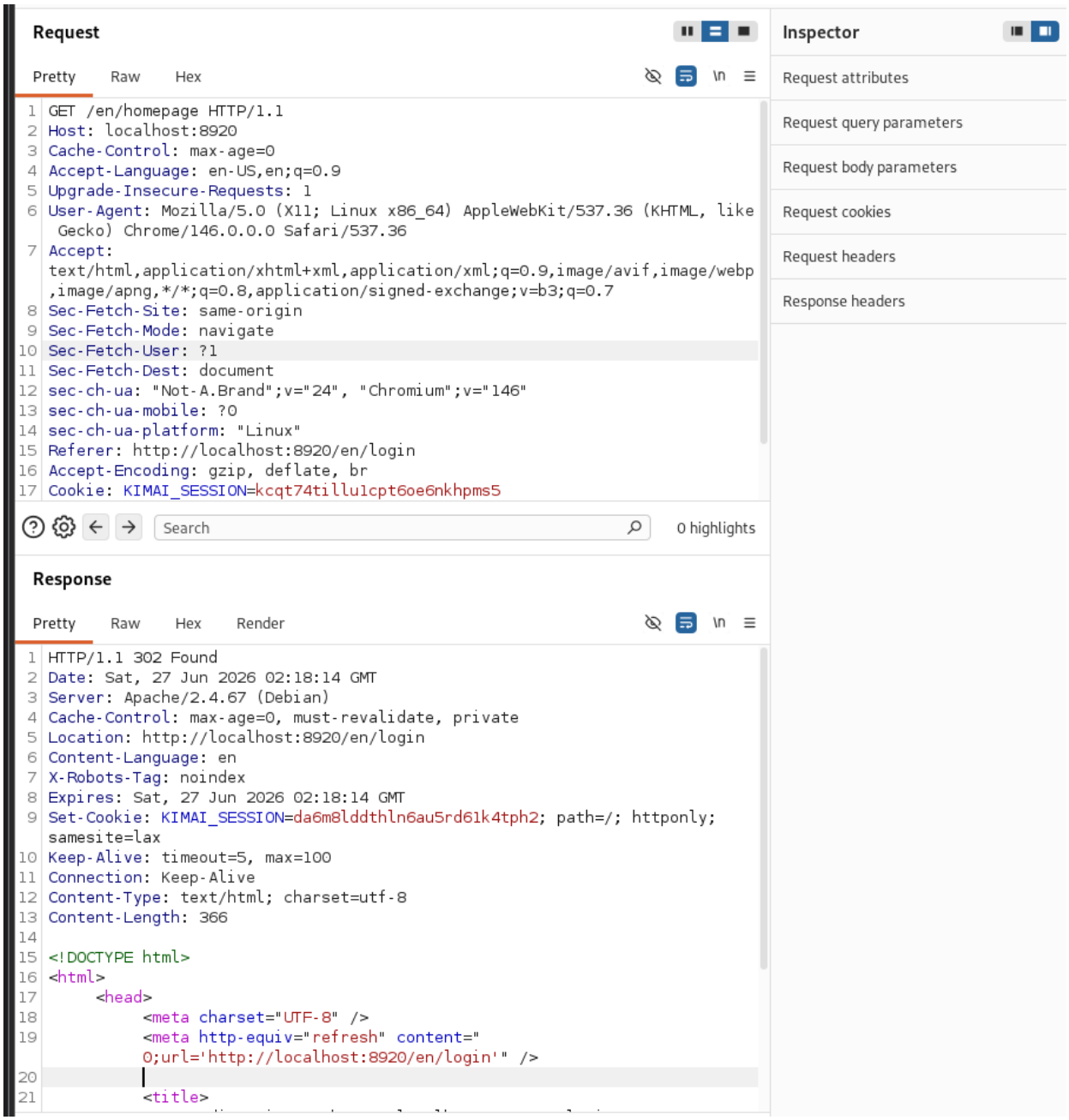


Figure 10 Burp Repeater showing the same KIMAI_SESSION cookie rejected after the server-side session row was deleted, returning HTTP 302 redirect to /en/login.

Figure 10 shows below deletion, the captured KIMAI_SESSION cookie was accepted and redirected to the authenticated timesheet page. After the matching row was deleted from kimai2_sessions, the same request was redirected to /en/login. This provides behavioral evidence that an invalidated server-side session is rejected. However, this test does not prove the exact idle timeout duration; the 7-day session.gc_maxlifetime value remains a Low severity hardening observation. The gap analysis recommends configuring an explicit shorter idle timeout and verifying rejection after real inactivity.

2. Authorization Testing

2.1 Vertical Access Control Bypass (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control

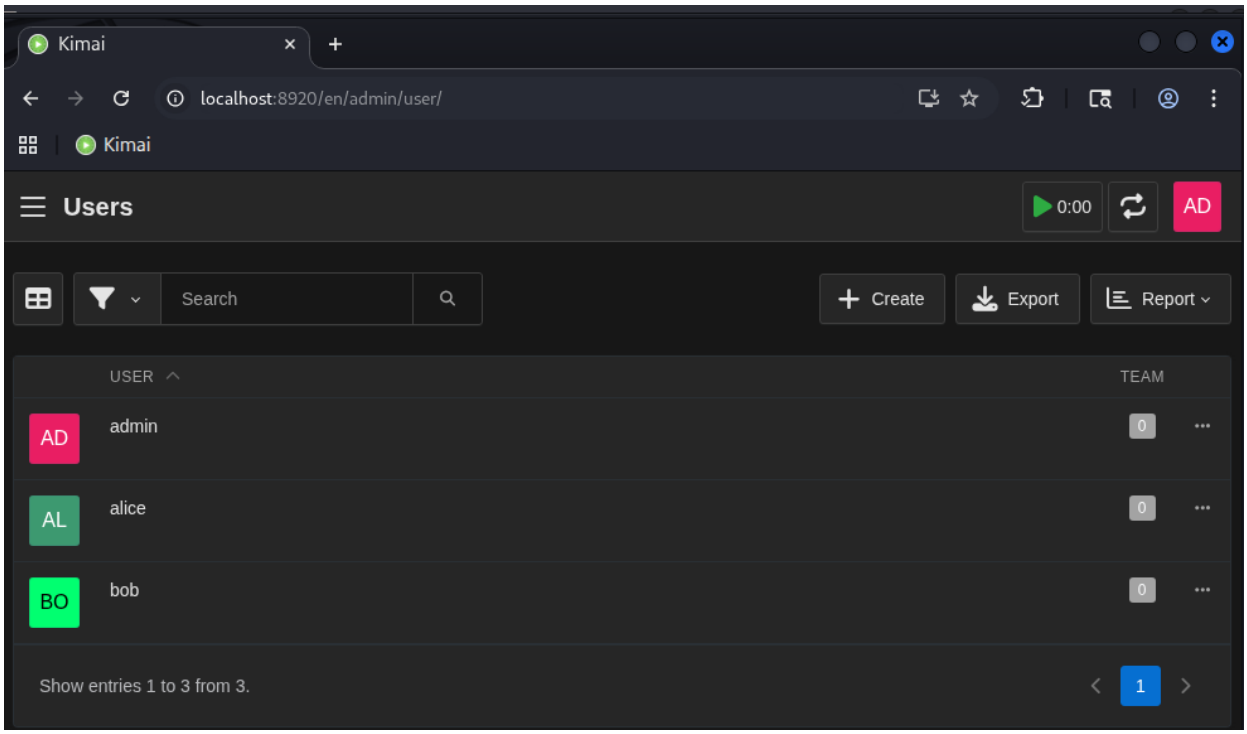


Figure 11 Admin-only Users management page at /en/admin/user/ accessed while logged in as admin, listing all three accounts.

Figure 11 shows while authenticated as admin, the System then Users menu opens the user management page at /en/admin/user/. This page lists all accounts and provides Create and Export controls, confirming it is an administrative function. This URL was used to test whether a standard user could reach the same page.

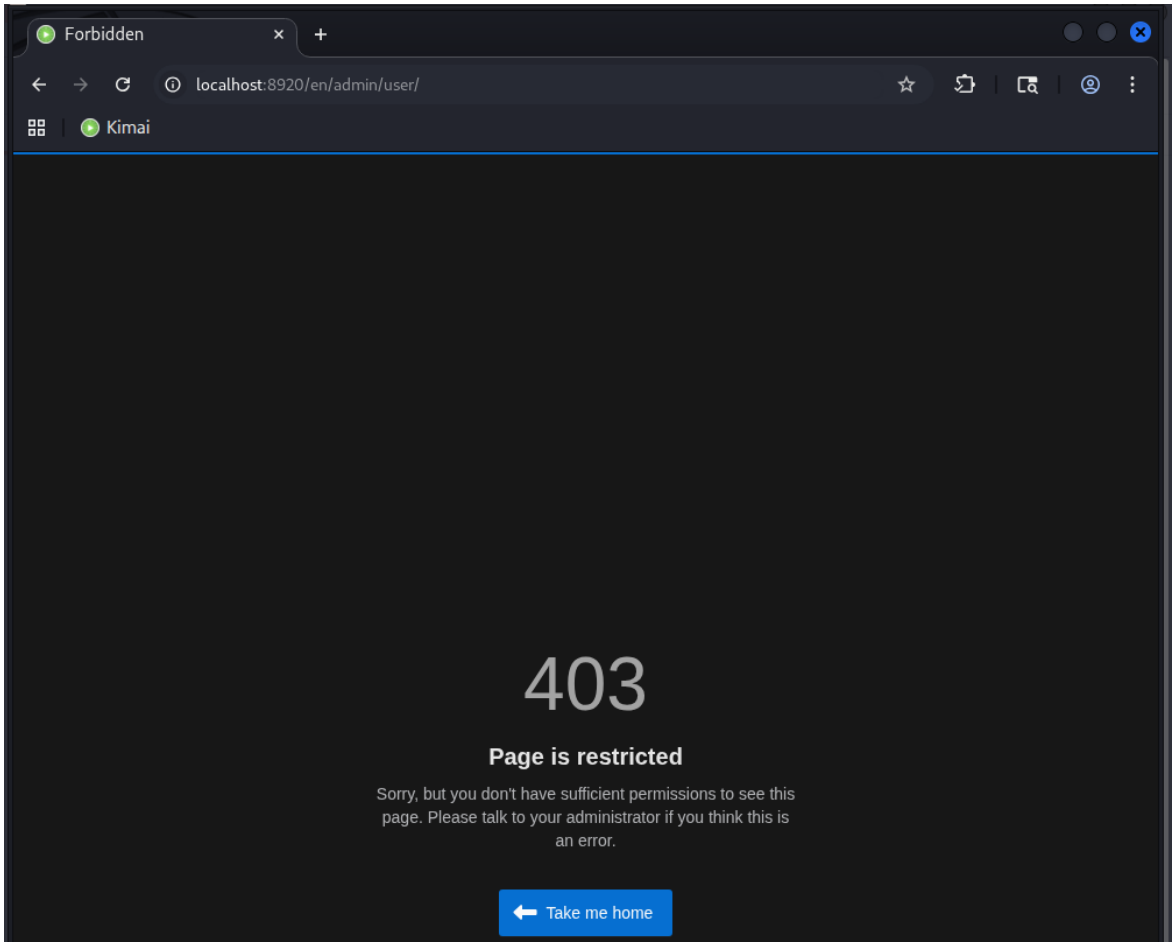


Figure 12 Standard user alice receiving HTTP 403 'Page is restricted' when requesting the admin-only Users URL directly.

Figure 12 shows after logging in as the standard user alice, the admin-only Users management URL was requested directly through the browser address bar. Kimai returned HTTP 403 Forbidden with the message 'Page is restricted.' This shows the authorization check is enforced at the server level rather than only hidden in the menu. Severity Informational, positive evidence of a correctly implemented control.

2.2 Horizontal Access Control Bypass (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control

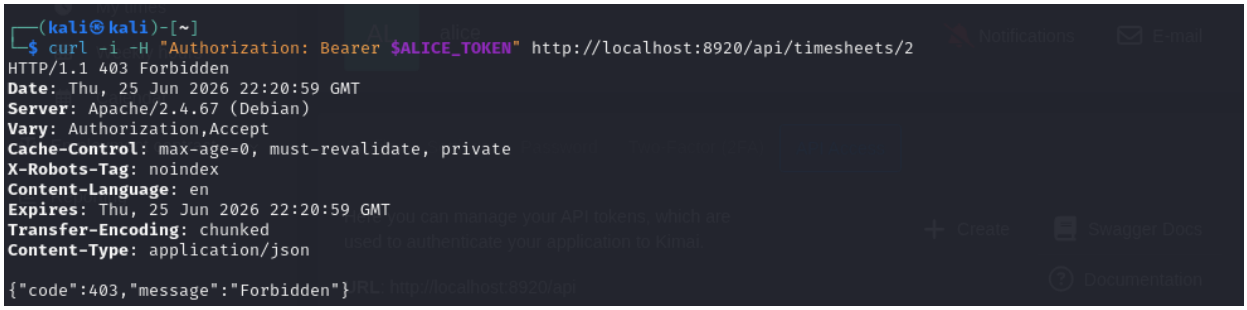


Figure 13 Standard user alice's API token receiving HTTP 403 Forbidden when requesting bob's timesheet entry via GET /api/timesheets/2.

Figure 13 shows alice's own API token, a request was sent to the timesheet endpoint for entry ID 2, which belongs to bob. Kimai returned HTTP 403 Forbidden, so alice's token was not able to read bob's timesheet. The authorization check is enforced at the API level.



Figure 14 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to create a new super-admin user via POST /api/users.

Figure 14 shows u sing alice's API token, a well-formed request was sent to create a new user with the ROLE_SUPER_ADMIN role. Kimai returned HTTP 403 Forbidden and no user was created. A standard user cannot invoke an administrative action by calling the endpoint directly, even with a complete and valid request body.

2.3 Special Request Header Handling (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control



Figure 15 Identical HTTP 403 response when alice's denied request is replayed with spoofed override and proxy headers (X-Original-URL, X-Rewrite-URL, X-Forwarded-For, X-Remote-IP, X-Client-IP).

Figure 15 shows Alice's previously denied request for bob's timesheet was resent with X-Original-URL and X-Rewrite-URL pointing at the target, and with X-Forwarded-For, X-Remote-IP, and

X-Client-IP set to 127.0.0.1 to imitate a trusted local origin. Kimai returned the same HTTP 403 Forbidden response. The headers produced no change in behavior, so the authorization decision is based on the authenticated user's role rather than on client-supplied headers.

2.4 Resources Assigned to a Different Role (WSTG-ATHZ-02) - OWASP A01:2025 Broken Access Control



Figure 16 Standard user alice's API token receiving HTTP 200 OK with a read-only customer list from GET /api/customers, confirming expected read visibility.

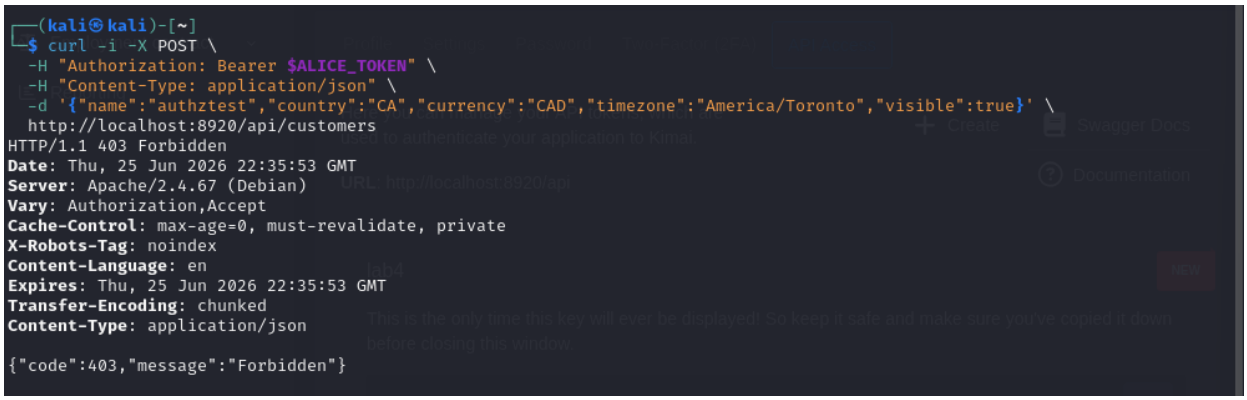


Figure 17 The same alice token receiving HTTP 403 Forbidden when attempting to create a customer via POST /api/customers, confirming write operations remain restricted.

Figure 17 shows Alice's token received HTTP 200 OK on GET /api/customers, returning customer records. In Kimai, standard users need read access to master data such as customers to select them during time tracking, so this is expected behavior rather than a bypass. When alice's token attempted to create a customer via POST, Kimai returned HTTP 403 Forbidden and no customer was created. Customer write operations remain restricted to higher roles. Severity Informational across all authorization tests.

2.5 Privilege Escalation (WSTG-ATHZ-03) - OWASP A01:2025 Broken Access Control

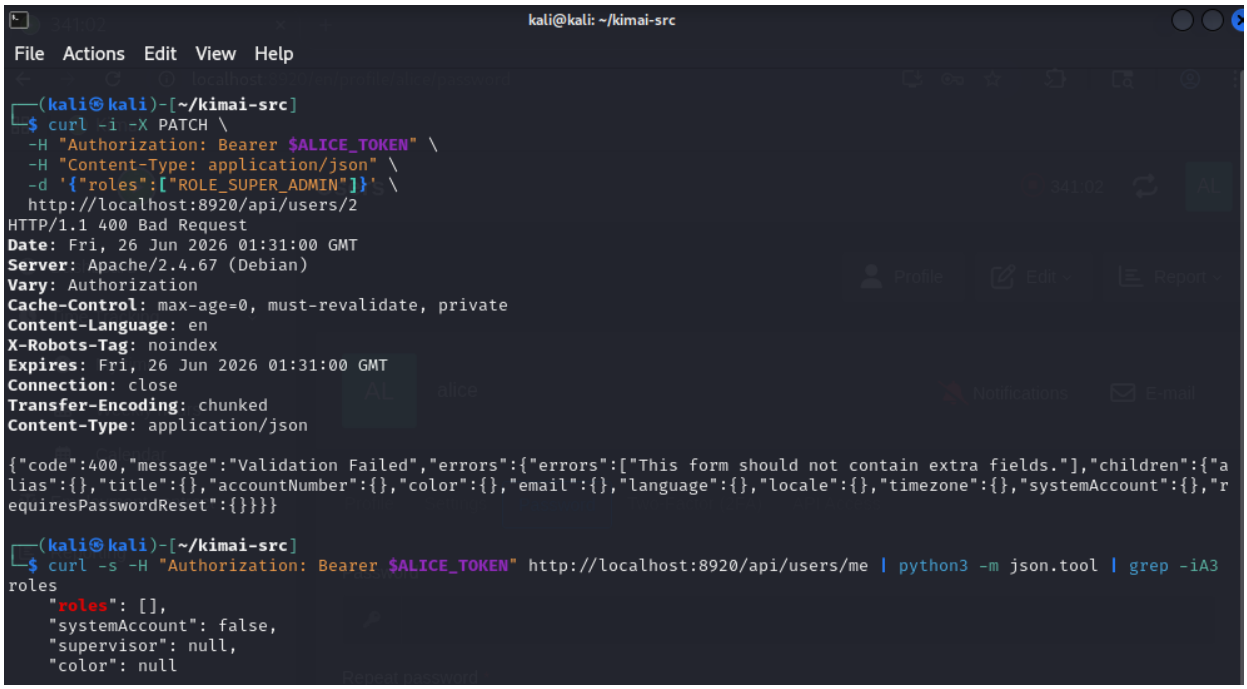


Figure 18 Standard user alice attempting to grant herself ROLE_SUPER_ADMIN via PATCH /api/users/2, rejected with HTTP 400 because the roles field is not part of her editable form.

Figure 18 shows WSTG 4.5.3 tests whether a user can manipulate their own role. Using alice's API token, a PATCH request to her own user record attempted to set her roles to ROLE_SUPER_ADMIN. The API rejected the request with HTTP 400 Bad Request and the message 'This form should not contain extra fields.' The roles field is not exposed in the user-edit form for a standard user. A follow-up request to /api/users/me confirmed her roles remained unchanged. No privilege escalation was achieved. Severity Informational.

3. Secret Management Testing

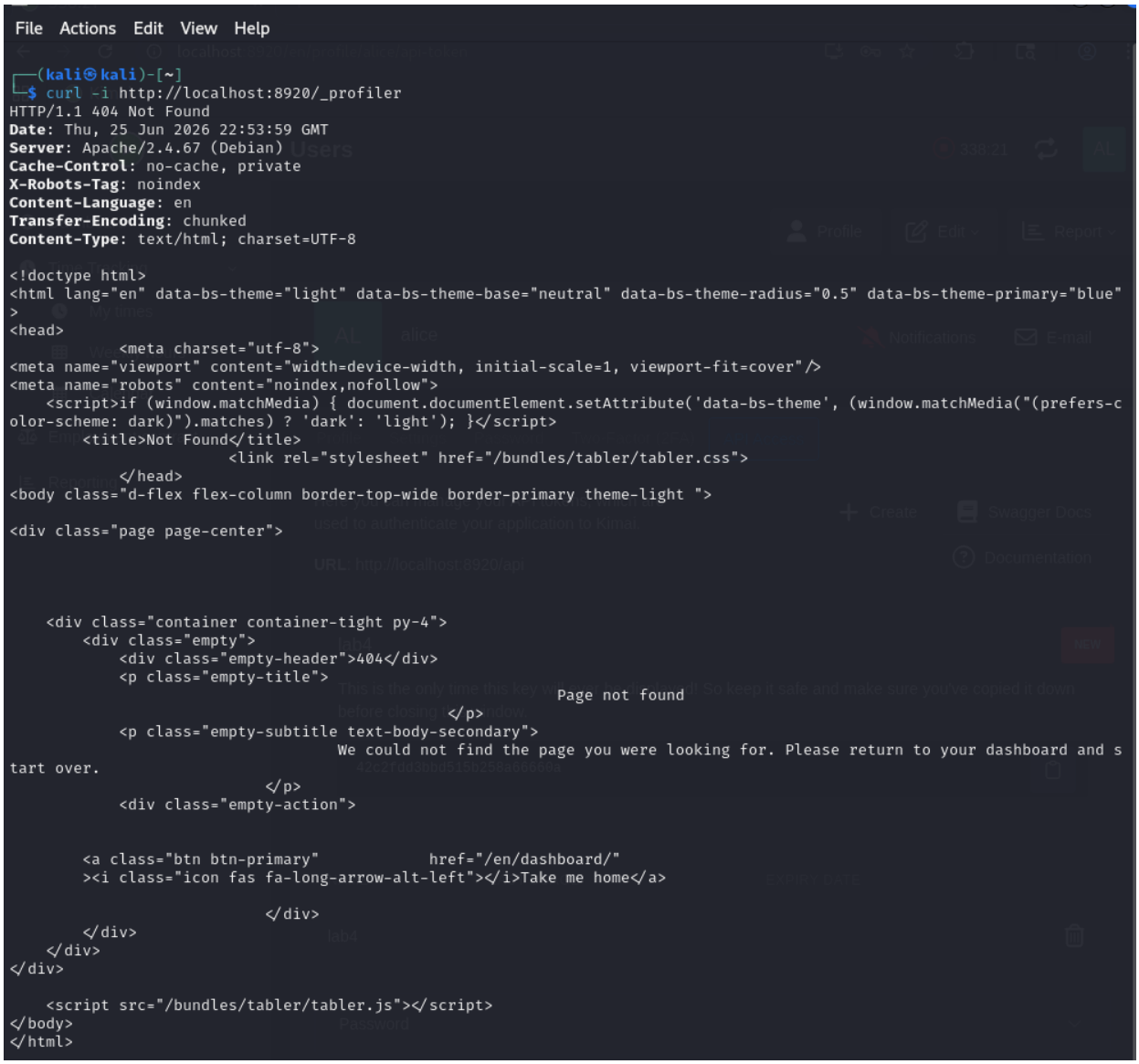
3.1 Storage of Application Secrets (WSTG-CONF-02) - OWASP A02:2025 Security Misconfiguration



Figure 19 Kimai container environment variables showing ADMINPASS and DATABASE_URL stored in plaintext, with APP_ENV correctly set to prod.

Figure 19 shows running container's environment shows the administrative password (ADMINPASS=changemeplease) and the full database connection string including credentials (DATABASE_URL=mysql://kimai:kimai@database:3306/kimai) held as plaintext environment variables, readable by any process inside the container. Storing the admin password and database credentials as plaintext environment variables is standard Docker behavior but remains a Low severity hardening gap, mapping to OWASP A02:2025 Security Misconfiguration.

3.2 Debug and Development Artifacts (WSTG-CONF-02) - OWASP A02:2025 Security Misconfiguration



```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ curl -i http://localhost:8920/_profiler
HTTP/1.1 404 Not Found
Date: Thu, 25 Jun 2026 22:53:59 GMT
Server: Apache/2.4.67 (Debian)
Cache-Control: no-cache, private
X-Robots-Tag: noindex
Content-Language: en
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

<!doctype html>
<html lang="en" data-bs-theme="light" data-bs-theme-base="neutral" data-bs-theme-radius="0.5" data-bs-theme-primary="blue"
>
<head>
        <meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, viewport-fit=cover"/>
<meta name="robots" content="noindex,nofollow">
    <script>if (window.matchMedia) { document.documentElement.setAttribute('data-bs-theme', (window.matchMedia("(prefers-c
olor-scheme: dark)").matches) ? 'dark': 'light'); }</script>
        <title>Not Found</title>
                        <link rel="stylesheet" href="/bundles/tabler/tabler.css">
        </head>
<body class="d-flex flex-column border-top-wide border-primary theme-light ">

<div class="page page-center">



    <div class="container container-tight py-4">
        <div class="empty">
            <div class="empty-header">404</div>
            <p class="empty-title">
                                                Page not found
                                            </p>
            <p class="empty-subtitle text-body-secondary">
                                    We could not find the page you were looking for. Please return to your dashboard and s
tart over.
                                </p>
            <div class="empty-action">


        <a class="btn btn-primary"            href="/en/dashboard/"
        ><i class="icon fas fa-long-arrow-alt-left"></i>Take me home</a>

                                </div>
        </div>
    </div>
</div>

    <script src="/bundles/tabler/tabler.js"></script>
</body>
</html>
```

Figure 20 Symfony debug profiler request returning HTTP 404 Not Found, confirming the development profiler is disabled in production.

Figure 20 shows a request to the Symfony debug profiler path (/_profiler) returned HTTP 404 Not Found, displaying only the standard application error page. This confirms the profiler is not reachable, consistent with the APP_ENV=prod setting. Severity Informational, positive evidence.

3.3 Secrets in Container Logs (WSTG-CONF-02) - OWASP A02:2025 / A09:2025

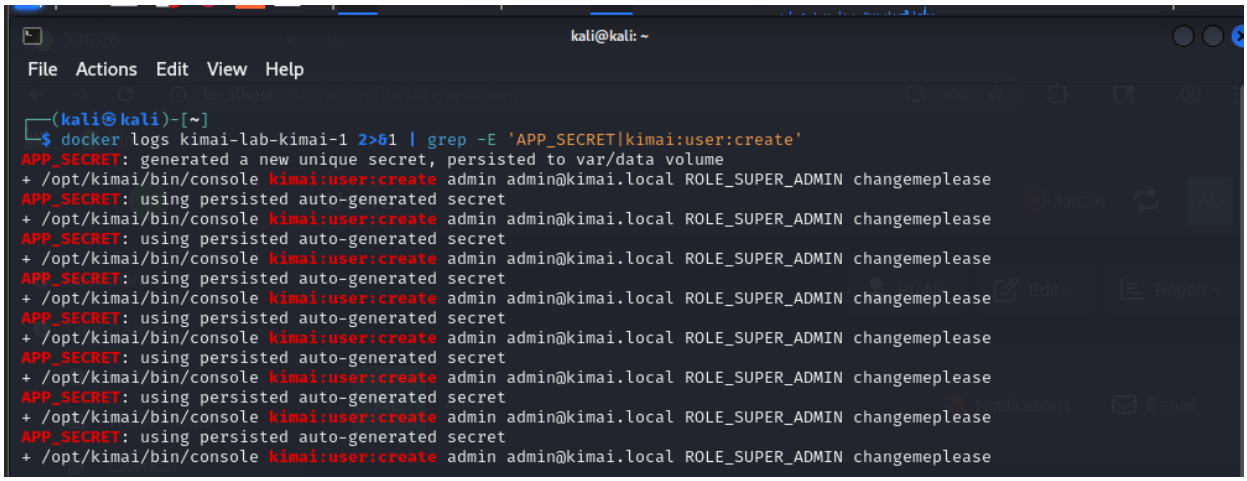


Figure 21 Container startup logs repeatedly printing the admin user-creation command with the plaintext password; APP_SECRET shown only as 'auto-generated' without its value.

Figure 21 shows the container's startup logs include the full administrative account creation command on every boot, including the plaintext password (kimai:user:create admin admin@kimai.local ROLE_SUPER_ADMIN changemeplease). The APP_SECRET encryption key is handled more safely: it is auto-generated once and reused from a persisted volume without its value being printed. Logging the admin password in cleartext is a Low severity hardening gap introduced by the container image's entrypoint script. It maps primarily to OWASP A02:2025 Security Misconfiguration, with OWASP A09:2025 Security Logging and Monitoring Failures as a secondary concern.

3.4 API Token Storage (WSTG-CONF-02) - OWASP A04:2025 Cryptographic Failures

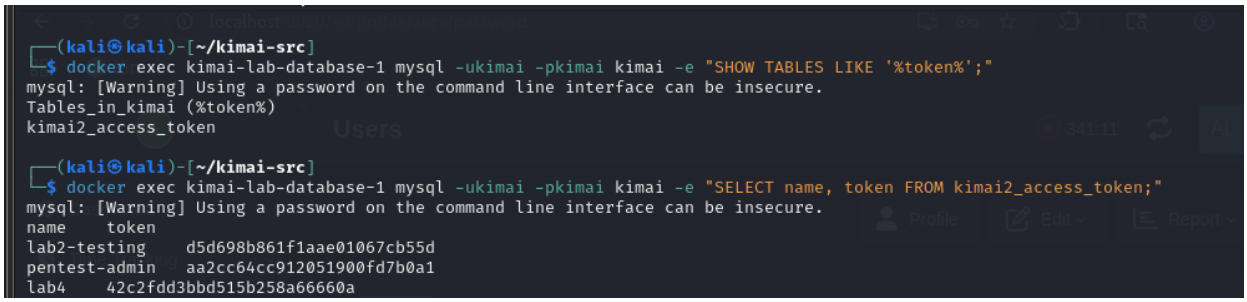


Figure 22 The kimai2_access_token table showing API tokens stored in plaintext, with the lab4 token matching the value used for API authentication.

Figure 22 shows kimai2_access_token table stores API tokens in plaintext. The lab4 token value in the database matches exactly the plaintext token used to authenticate API requests throughout this lab, confirming tokens are not hashed or encrypted at rest. Any party with read access to the database can recover every user's API token and replay it. This contrasts with user passwords, which are bcrypt-hashed and cannot normally be replayed directly. This is a Medium severity weakness mapping to OWASP A04:2025 Cryptographic Failures. The recommended control is to hash tokens at rest using a suitable one-way function and compare hashes on authentication.

3.5 Encryption Key (APP_SECRET) Management (WSTG-CONF-02) - OWASP A04:2025 Cryptographic Failures

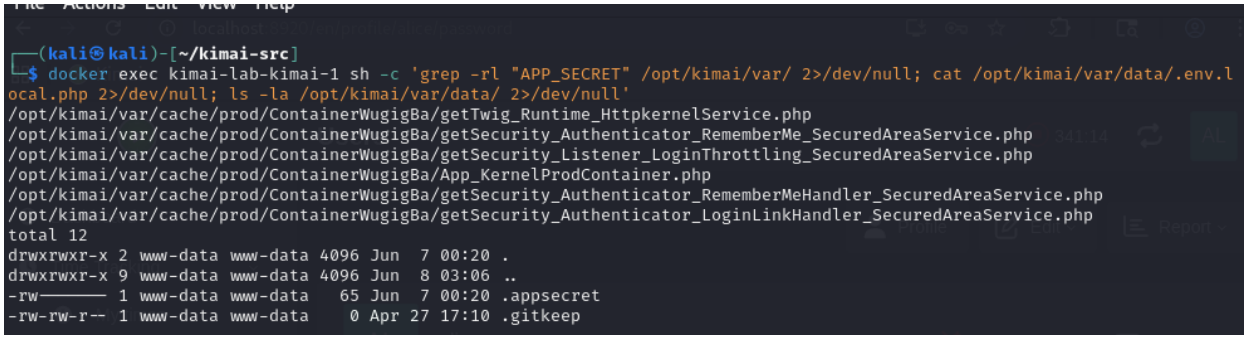


Figure 23 APP_SECRET stored in /opt/kimai/var/data/.appsecret with owner-only permissions (-rw-------) and present in the production cache.

Figure 23 shows the APP_SECRET encryption key is persisted to /opt/kimai/var/data/.appsecret with file permissions of 600, meaning only the application's www-data owner can read it. The data directory is not web-accessible. The key also appears in the compiled production cache, which is expected Symfony behavior where configuration is baked into the container cache; these files are internal and not externally exposed. The key is therefore stored with appropriate access restrictions. Severity Informational, positive evidence.

3.6 Weak Encryption and Password Hashing (WSTG-CRYP-04) - OWASP A04:2025 Cryptographic Failures

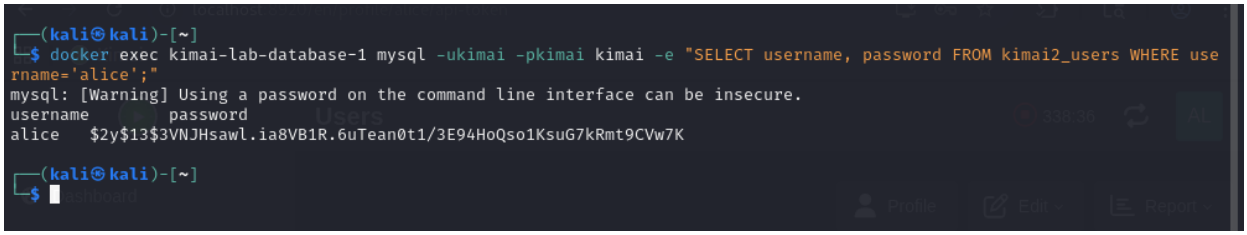


Figure 24 Stored password hash for the alice account showing bcrypt with cost factor 13 ($2y$13$), confirming a strong adaptive hashing algorithm.

Figure 24 shows the stored password hash for the alice account begins with $2y$13$, identifying the algorithm as bcrypt with a work factor of 13. Bcrypt is an approved password-hashing function under WSTG, and cost 13 is above the common default of 12, increasing resistance to brute-force attacks. Severity Informational, positive evidence.

3.7 Source Code Review for Weak Algorithms (WSTG-CRYP-04) - OWASP A04:2025 Cryptographic Failures



Figure 25 Recursive source search of Kimai's PHP src/ directory for weak cryptographic algorithm keywords returning no matches.

Figure 25 shows Kimai's own application source (the src/ directory of version 2.59.0) was searched recursively for weak-algorithm keywords: MD5, MD4, SHA1, RC4, RC2, Blowfish, DES, and ECB. The search returned no matches. This does not cover Composer or JavaScript dependencies, wrapper functions, or dynamically selected algorithms. Combined with the bcrypt password hashing confirmed above, the available evidence is consistent with the application relying on strong cryptographic primitives. Severity Informational.

4. API Security Testing

4.1 Insecure Direct Object References / BOLA (WSTG 4.5.4) - OWASP A01:2025 Broken Access Control

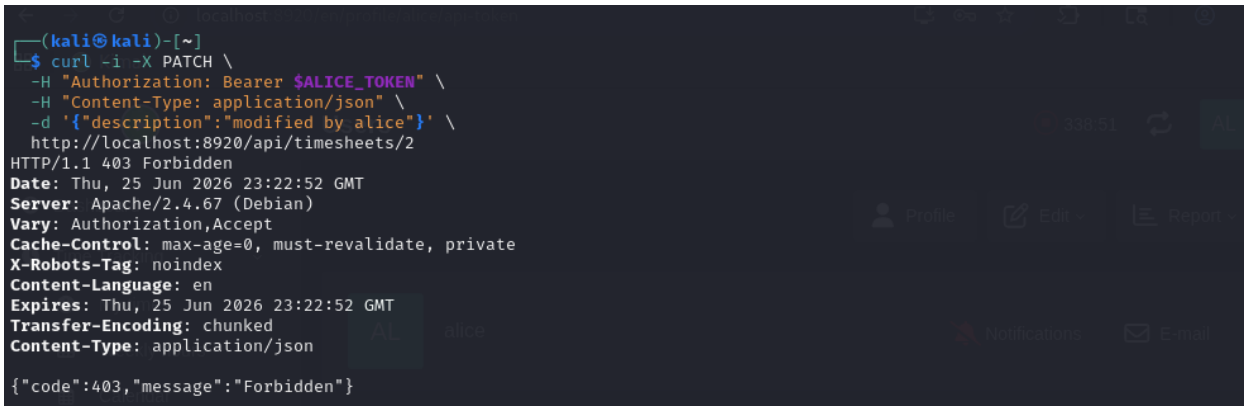


Figure 26 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to modify bob's timesheet entry via PATCH /api/timesheets/2.

Figure 26 shows this test checked whether the timesheet ID can be used as an insecure direct object reference to alter another user's record. Using alice's API token, a PATCH request attempted to change the description field of timesheet entry ID 2, which belongs to bob. Kimai returned HTTP 403 Forbidden with body {"code":403,"message":"Forbidden"}, and no change was made. The object-level authorization check is enforced on write operations, so a standard user cannot modify another user's data by supplying its ID. This is a pass with no finding, mapping to OWASP A01:2025 Broken Access Control, severity Informational.

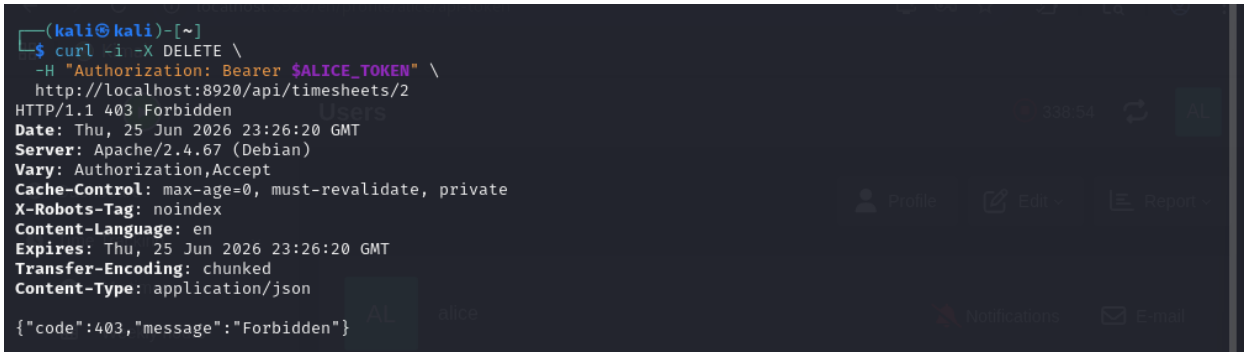


Figure 27 Standard user alice's API token receiving HTTP 403 Forbidden when attempting to delete bob's timesheet entry via DELETE /api/timesheets/2.

Figure 27 shows this test checked whether the timesheet ID could be used as an insecure direct object reference to delete another user's record. Using alice's API token, a DELETE request targeted timesheet entry ID 2, which belongs to bob. Kimai returned HTTP 403 Forbidden with body {"code":403,"message":"Forbidden"}, and the entry was not removed. Object-level authorization is enforced on delete operations as well, so a standard user cannot destroy another

user's data by supplying its ID. This is a pass with no finding, mapping to OWASP A01:2025 Broken Access Control, severity Informational.

Insecure direct object reference testing assessed whether a standard user could act on another user's objects by manipulating the object ID in API requests. Using alice's API token against timesheet entry ID 2 (owned by bob), all three operations were rejected with HTTP 403 Forbidden: read (tested in the authorization area), modify (PATCH), and delete (DELETE). Object-level authorization is consistently enforced across read and write paths at the API level, so no IDOR weakness was identified. This maps to OWASP A01:2025 Broken Access Control as positive evidence of a correctly implemented control, with an overall severity of Informational.

4.2 API Authentication Rate Limiting (WSTG-ATHN-03) - OWASP A07:2025 Authentication Failures

```
┌──(kali㉿kali)-[~]
└─$ for i in $(seq 1 30); do
  curl -s -o /dev/null \
    -w "$i | HTTP %{http_code} | time %{time_total}s\n" \
    -H "Authorization: Bearer $ALICE_TOKEN" \
    "http://localhost:8920/api/users/me"
done
1 | HTTP 200 | time 0.020709s
2 | HTTP 200 | time 0.011544s
3 | HTTP 200 | time 0.011289s
4 | HTTP 200 | time 0.010094s
5 | HTTP 200 | time 0.010241s
6 | HTTP 200 | time 0.009596s
7 | HTTP 200 | time 0.009944s
8 | HTTP 200 | time 0.009804s
9 | HTTP 200 | time 0.010593s
10 | HTTP 200 | time 0.009772s
11 | HTTP 200 | time 0.010878s
12 | HTTP 200 | time 0.011072s
13 | HTTP 200 | time 0.010148s
14 | HTTP 200 | time 0.010542s
15 | HTTP 200 | time 0.010013s
16 | HTTP 200 | time 0.010975s
17 | HTTP 200 | time 0.010144s
18 | HTTP 200 | time 0.009997s
19 | HTTP 200 | time 0.011095s
20 | HTTP 200 | time 0.009919s
21 | HTTP 200 | time 0.009021s
22 | HTTP 200 | time 0.010319s
23 | HTTP 200 | time 0.010239s
24 | HTTP 200 | time 0.010820s
25 | HTTP 200 | time 0.009915s
26 | HTTP 200 | time 0.010109s
27 | HTTP 200 | time 0.011902s
28 | HTTP 200 | time 0.010560s
29 | HTTP 200 | time 0.010536s
30 | HTTP 200 | time 0.009780s

┌──(kali㉿kali)-[~]
└─$
```

Figure 28 Thirty repeated authenticated requests to /api/users/me using Alice's valid token returning HTTP 200 with no throttling observed.

A bounded rate-limiting test was performed by sending 30 repeated authenticated requests to the /api/users/me endpoint using Alice’s valid API token. All 30 requests returned HTTP 200 OK,

and no HTTP 429 Too Many Requests response was observed. The response times also stayed consistent during the test. This does not prove that rate limiting is absent in all situations, but it shows that no throttling was observed during this local 30-request authenticated API test.

```
┌──(kali㉿kali)-[~]
└─$ for i in $(seq 1 30); do
  curl -s -o /dev/null \
    -w "$i | HTTP %{http_code} | time %{time_total}s\n" \
    -H "Authorization: Bearer invalid-token-$i" \
    "http://localhost:8920/api/users/me"
done
1 | HTTP 401 | time 0.009853s
2 | HTTP 401 | time 0.005073s
3 | HTTP 401 | time 0.006053s
4 | HTTP 401 | time 0.005275s
5 | HTTP 401 | time 0.005277s
6 | HTTP 401 | time 0.005471s
7 | HTTP 401 | time 0.005975s
8 | HTTP 401 | time 0.005658s
9 | HTTP 401 | time 0.005471s
10 | HTTP 401 | time 0.004926s
11 | HTTP 401 | time 0.005866s
12 | HTTP 401 | time 0.005397s
13 | HTTP 401 | time 0.005640s
14 | HTTP 401 | time 0.005908s
15 | HTTP 401 | time 0.005393s
16 | HTTP 401 | time 0.005542s
17 | HTTP 401 | time 0.006161s
18 | HTTP 401 | time 0.006182s
19 | HTTP 401 | time 0.006875s
20 | HTTP 401 | time 0.005518s
21 | HTTP 401 | time 0.005194s
22 | HTTP 401 | time 0.005878s
23 | HTTP 401 | time 0.005822s
24 | HTTP 401 | time 0.005523s
25 | HTTP 401 | time 0.005612s
26 | HTTP 401 | time 0.005514s
27 | HTTP 401 | time 0.006000s
28 | HTTP 401 | time 0.006089s
29 | HTTP 401 | time 0.005403s
30 | HTTP 401 | time 0.005458s

┌──(kali㉿kali)-[~]
└─$
```

Figure 29 Thirty repeated invalid API-token requests returning HTTP 401 without throttling or delay increase.

Figure 29 shows a bounded rate-limiting test sent 30 repeated authenticated requests to /api/users/me using alice's valid token. All 30 returned HTTP 200 OK with no HTTP 429 Too Many Requests observed. A second test used 30 different invalid bearer tokens against the same endpoint; all returned HTTP 401 Unauthorized with no throttling. This does not prove rate limiting

is absent in all situations, but no throttling was triggered during the tested patterns. The gap analysis recommends applying per-token and per-IP rate limiting to API endpoints. Severity Low.

4.3 Error Handling and Malformed Input (WSTG-ERRH-01) - OWASP A10:2025



Figure 30 Baseline PATCH request showing alice successfully updating her own timesheet entry via /api/timesheets/3 with HTTP 200 OK.

Figure 30 shows before testing malformed input, a valid baseline request was sent to confirm that the endpoint, API token, and object ID were working correctly. Alice's API token was used to send a PATCH request to her own timesheet entry, ID 3, with a normal description value. Kimai returned HTTP 200 OK and the response body showed the same timesheet ID, user ID 2, and description value. This confirms that later error responses are caused by the malformed input being tested, not by an invalid token, wrong endpoint, or lack of permission.



Figure 31 Kimai returning HTTP 400 Bad Request with structured validation error 'Please enter a valid date and time' when the begin field is set to 'not-a-date'.

Figure 31 shows confirming that a valid PATCH request to Alice's own timesheet entry worked, the begin field was changed to the invalid value "not-a-date". Kimai rejected the request with HTTP 400 Bad Request and returned a structured JSON validation error. The response identified the invalid begin field and stated "Please enter a valid date and time." No stack trace, SQL error, file path, or debug details were exposed in the response body. This shows that this malformed input was handled through input validation rather than an unhandled server error.



Figure 32 Follow-up GET request confirming the malformed PATCH did not change alice's timesheet; original values remain.

Figure 32 shows a follow-up GET request was sent to the same timesheet entry after the malformed PATCH request was rejected. Kimai returned HTTP 200 OK, and the response still showed the original begin time, end time, description, and timesheet ID. This confirms that the invalid begin value was rejected without modifying the record. Combined with the previous HTTP 400 validation response, this supports that the tested malformed input was handled safely and did not cause an unintended state change.



Figure 33 Container log review showing the malformed PATCH returned HTTP 400 without a related exception or stack trace.

Figure 33 shows confirming a valid baseline PATCH worked, the begin field was changed to the invalid value 'not-a-date'. Kimai rejected the request with HTTP 400 and a structured JSON validation error identifying the problem field. No stack trace, SQL error, internal file path, or debug details appeared in the response body or container logs, and a follow-up GET confirmed the record was not modified. The malformed input was handled through validation rather than an unhandled server error. Severity Informational.

4.4 SQL Injection Testing (WSTG-INPV-05) - OWASP A05:2025 Injection



Figure 34 Single-quote injection in the term parameter of GET /api/timesheets returning HTTP 200 with an empty result set and no SQL error.

Figure 34 shows WSTG 4.7.5 begins by injecting a single quote into each parameter that may feed a database query and watching for a resulting SQL error. The free-text term search parameter was selected as the most likely injectable input. A request with term=test' returned HTTP 200 with an empty JSON array and no error message, stack trace, or database error in the response. The single quote did not break the query or produce a 500 error, indicating no error-based SQL injection was observed in the tested term parameter with this probe. This maps to OWASP A05:2025 Injection.

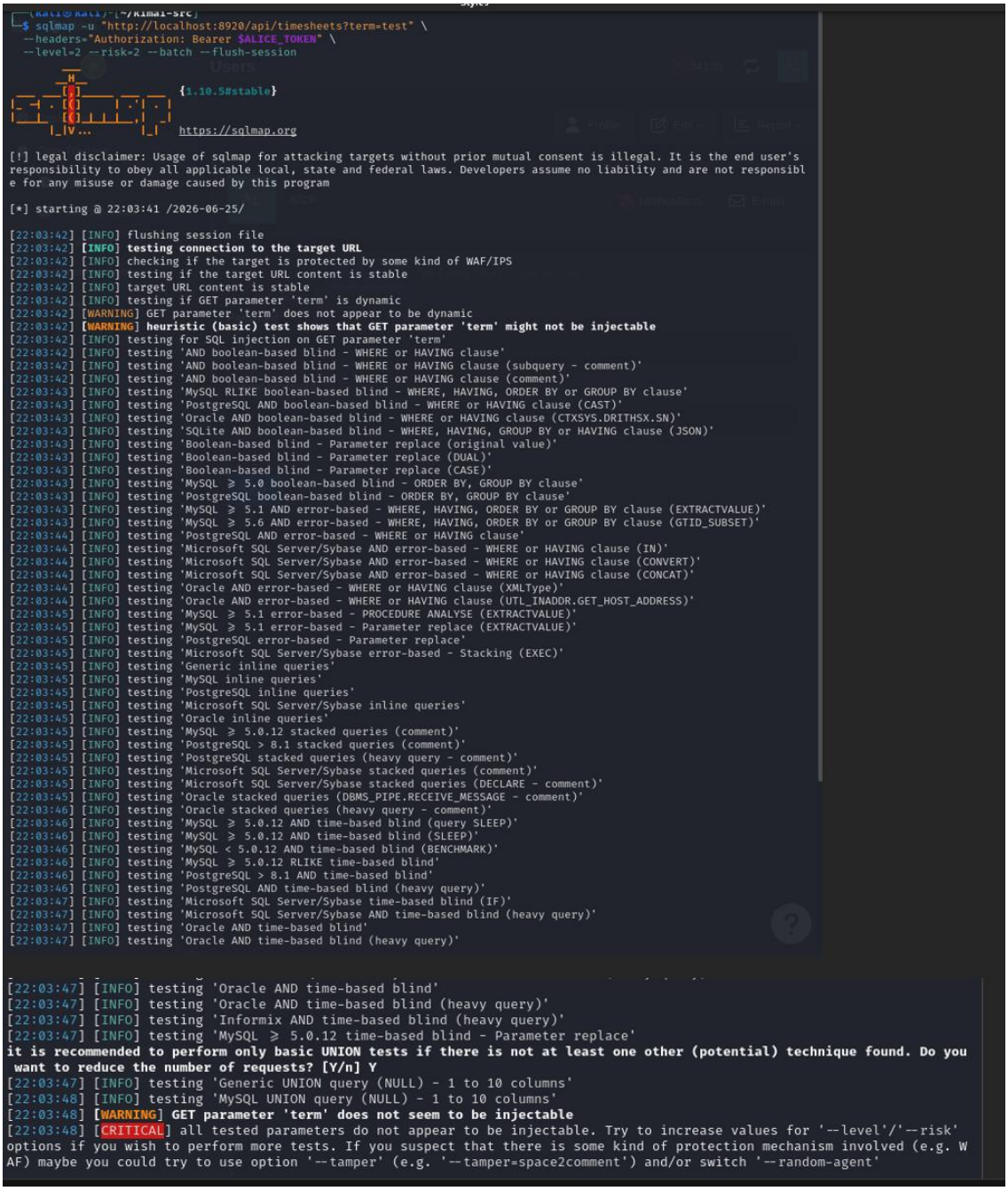


Figure 35 sqlmap testing the term parameter at level 2 / risk 2 against the Kimai API, concluding the parameter does not appear to be injectable.

Figure 35. sqlmap testing the term parameter at level 2 / risk 2 against the Kimai API, concluding the parameter does not appear to be injectable.

Figure 35 Following WSTG 4.7.5's direction to use automated tooling for thorough injection testing, sqlmap was run against the term parameter of GET /api/timesheets using alice's API token for authentication, at level 2 and risk 2. sqlmap tested the parameter across its full range of techniques, including boolean-based blind, error-based, stacked queries, time-based blind, and UNION queries, against MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and SQLite payload sets. It concluded that the term parameter does not appear to be injectable across the tested payload set. No SQL injection was identified in the tested term parameter using the selected manual and sqlmap payloads. This result is consistent with safe query handling, but a negative black-box test does not independently prove that every related query in the application is parameterised. This maps to OWASP A05:2025 Injection, severity Informational.

SQL injection testing combined a manual probe and automated scanning against the API's free-text term search parameter. A single-quote injection returned a clean HTTP 200 with no SQL error, and sqlmap testing at level 2 risk 2 across boolean, error-based, stacked, time-based, and UNION techniques found no SQL injection in the tested endpoint, parameter, and payload set. The result is consistent with safe query handling, though it does not prove every related query is parameterised. This maps to OWASP A05:2025 Injection as positive evidence of a correctly implemented control, with an overall severity of Informational.

5. Fuzzing

5.1 API Parameter Fuzzing (Timesheet ID) - OWASP A01:2025 Broken Access Control

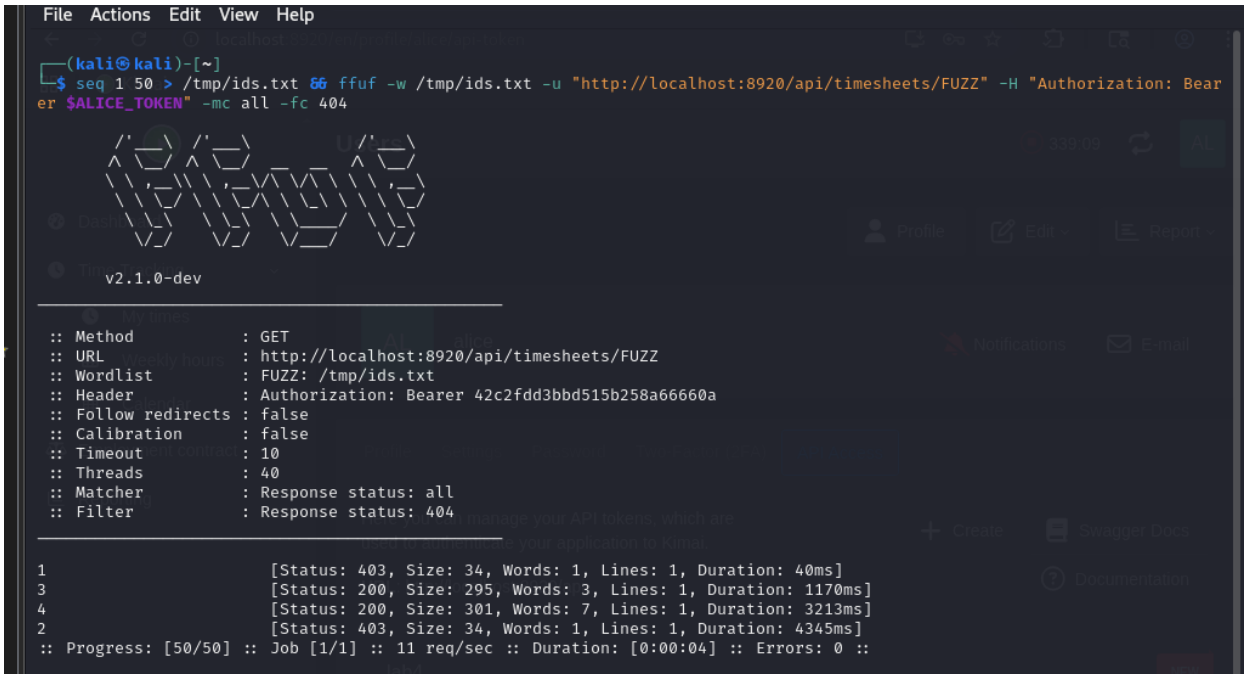


Figure 36 ffuf fuzzing the timesheet ID parameter from 1 to 50 using alice's token, returning HTTP 200 only for alice's entries (IDs 3 and 4) and HTTP 403 for other users' entries.

Figure 36 shows this test applied fuzzing to an API parameter, one of the required fuzzing targets, using ffuf against the timesheet ID in GET /api/timesheets/{id}. A range of IDs from 1 to 50 was tested with alice's API token, with 404 responses filtered out to isolate the meaningful results. The fuzzer returned HTTP 200 for IDs 3 and 4, which belong to alice, and HTTP 403 for IDs 1 and 2, which belong to other users. This confirms at scale that the API applies object-level authorization based on ownership rather than returning any record whose ID is supplied, consistent with the manual IDOR results. The fuzzing demonstrated correct access control behaviour and revealed no unauthorized data exposure, mapping to OWASP A01:2025 Broken Access Control, severity Informational.

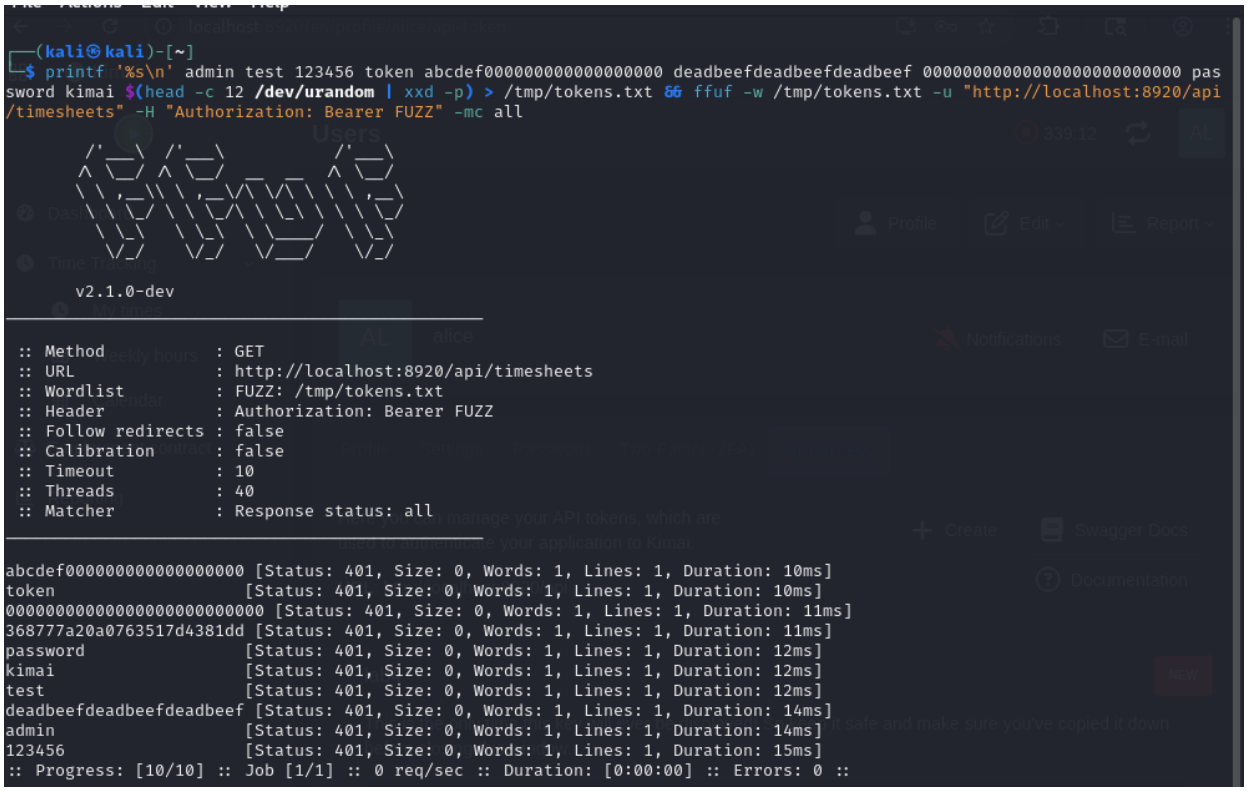


Figure 37 ffuf fuzzing the bearer token field with ten guessed and random values, all returning HTTP 401 Unauthorized.

Figure 37 shows applied fuzzing to the token field, one of the required fuzzing targets, using ffuf to submit a list of candidate bearer tokens in the Authorization header against GET /api/timesheets. The list included common guessable strings (admin, password, kimai, 123456, test) and random hex values. All ten returned HTTP 401 Unauthorized with an empty body, and none granted access. This shows that the API uniformly rejected the ten tested invalid token values. It does not by itself prove that every issued token is unpredictable or that no weak token could exist. No authentication bypass was identified in this bounded test, mapping to OWASP A07:2025 Authentication Failures, severity Informational.

5.2 Authentication Endpoint Fuzzing (Login Form) - OWASP A07:2025 Authentication Failures



Figure 38 Login page response showing 'Invalid credentials' after submitting alice's username with an incorrect password, confirming the request reached authentication logic.

Figure 38 shows submitting Alice's username with an incorrect password using a valid session cookie and CSRF token, the response redirected back to the login page. The returned login page displayed the message "Invalid credentials." This confirms that the request reached Kimai's authentication logic and was rejected because the password was incorrect, rather than being blocked by an invalid CSRF token before password validation.

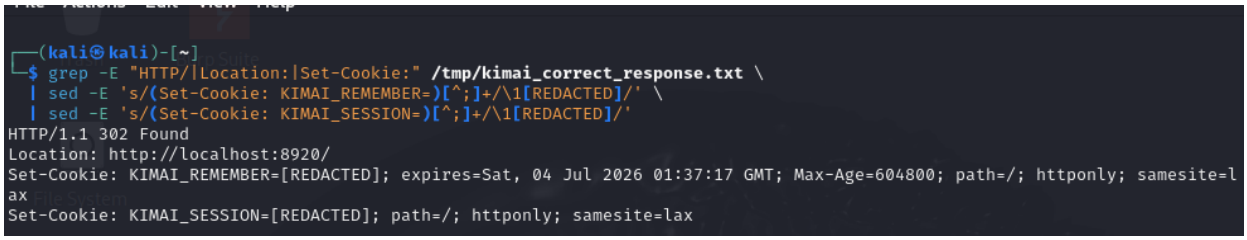


Figure 39 Successful login response confirming valid credentials redirect to the application root and issue authenticated session cookies.

Figure 39 shows controlled login-form test used a valid CSRF token and matching session cookie. An incorrect password was rejected with 'Invalid credentials,' while valid credentials redirected to the application root and issued authenticated session cookies. This confirms the test reached Kimai's authentication logic rather than being blocked by a stale CSRF token. No authentication bypass was identified. Severity Informational.

5.3 Secret-Related Input Fuzzing - OWASP A07:2025 Authentication Failures

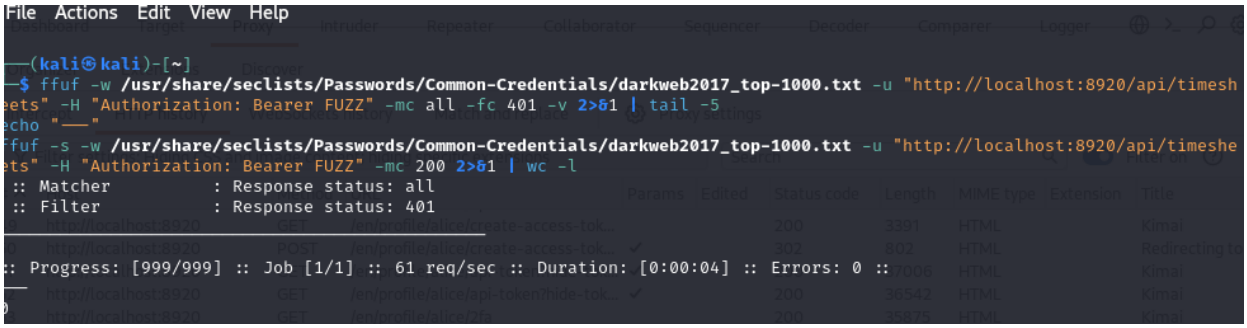


Figure 40 ffuf fuzzing the API bearer token with 999 values from the SecLists darkweb2017 leaked-credentials wordlist, with zero HTTP 200 responses.

Figure 40 shows ffuf submitted 999 values from the SecLists darkweb2017 leaked-credentials wordlist as the API bearer token against GET /api/timesheets. All 999 requests were rejected with HTTP 401 Unauthorized. None of the tested leaked-credential values matched a valid token in this deployment. It does not establish that every issued API token is unpredictable or that all possible weak values would be rejected. Severity Informational.

6. Static Code Analysis

6.1 Software Composition Analysis with Snyk - OWASP A03:2025 Software Supply Chain



Figure 41 Snyk dependency scan of Kimai 2.59.0 source reporting a Critical deserialization vulnerability in phpoffice/phpspreadsheet and a Medium remote-file-inclusion in mpdf/mpdf, along with Medium JavaScript issues.

Figure 41 shows Snyk to perform software composition analysis on the Kimai 2.59.0 source, scanning its declared dependencies against Snyk's vulnerability database. The scan covered 220 PHP dependencies (composer.lock) and 473 JavaScript dependencies (pnpm-lock.yaml). It reported a Critical deserialization-of-untrusted-data vulnerability in phpoffice/phpspreadsheet and a Medium remote-file-inclusion vulnerability in mpdf/mpdf, along with several Medium JavaScript issues, most notably cross-site scripting and initialization flaws in dompurify. These are known vulnerabilities in third-party libraries bundled by Kimai rather than flaws demonstrated to be exploitable in Kimai itself, since reachability depends on whether the application exposes the affected code paths to attacker input. The presence of outdated dependencies with known CVEs is a supply-chain risk and maps to OWASP A03:2025 Software Supply Chain Failures. Two severities are distinguished: the upstream advisory severity (up to Critical for phpoffice/phpspreadsheet) and the contextual severity for this Kimai deployment, which is undetermined because reachability of the affected code paths was not confirmed. The application is not rated Critical solely on the basis of the upstream advisory rating.

6.2 Static Analysis with Semgrep - OWASP A05:2025 Injection

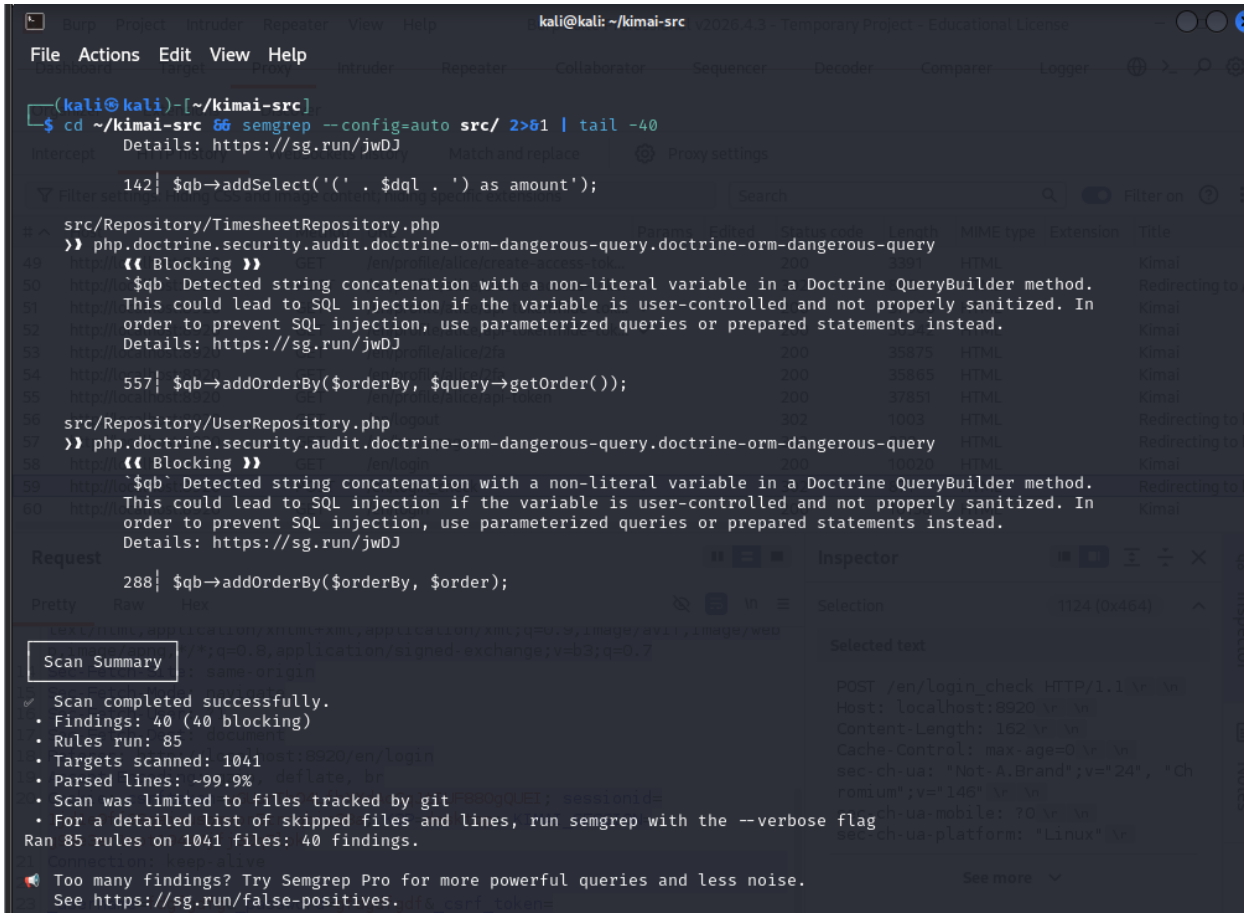


Figure 42 Semgrep scan of Kimai's PHP source (src/) running 85 rules across 1041 files, reporting 40 findings predominantly consisting of Doctrine QueryBuilder string-concatenation warnings.

Figure 42 shows Semgrep ran 85 security rules across 1041 PHP files in Kimai's src/ directory and returned 40 findings. The majority are raised by the Doctrine ORM dangerous-query rule, which

flags string concatenation of non-literal variables in query-builder methods such as addOrderBy and addSelect as potential SQL injection. Representative flagged lines were reviewed: the order-by and column variables trace to fixed internal allow-list values rather than attacker-controlled input, consistent with the Lab 3 sqlmap result of no SQL injection. These are assessed as likely false positives or unconfirmed findings requiring manual verification. Severity Informational to Low pending confirmation.

6.3 ESLint Security Plugin - OWASP A05:2025 Injection

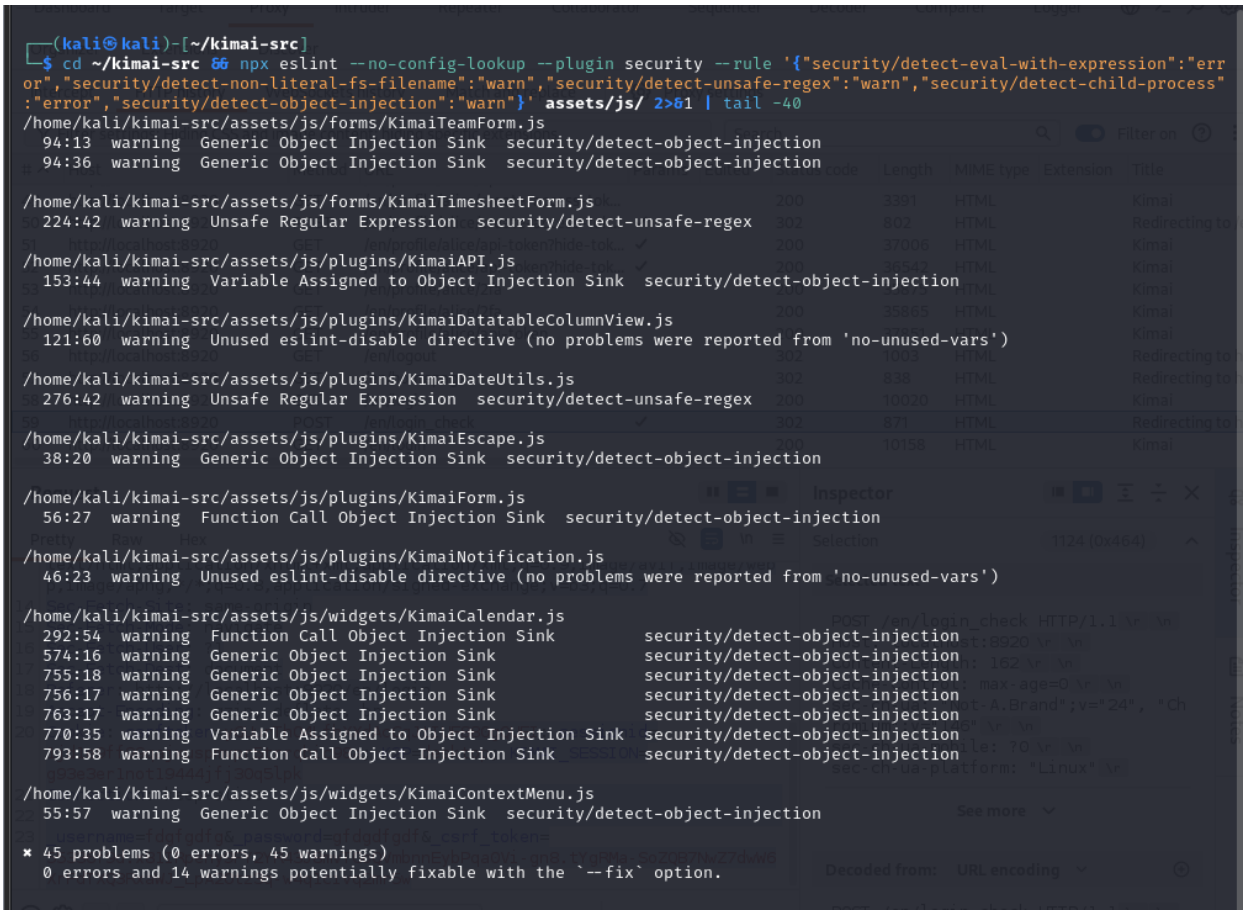


Figure 43 ESLint security-plugin scan of Kimai's JavaScript (assets/js/) reporting 45 warnings and 0 errors, with no eval or child-process patterns detected.

Figure 43 shows ESLint with the security plugin ran against Kimai's JavaScript source (assets/js/). The scan reported 45 warnings and 0 errors. The high-severity rules configured as errors, detection of eval-with-expression and child-process execution, produced no matches. The 45 warnings are almost entirely generic object-injection flags raised by detect-object-injection, which marks any bracket-notation property access and is known for a high false-positive rate, along with two unsafe-regular-expression warnings. Each warning would require input-to-sink tracing to classify definitively. Assessed as unconfirmed findings or likely false positives. Severity Informational to Low.

6.4 Tool Applicability: Bandit (Not Applicable)

Bandit, a static analysis tool listed in the lab toolset, was not executed because it analyzes Python source code exclusively. A search of the Kimai 2.59.0 source confirmed it contains no Python files; the codebase is PHP and JavaScript. Bandit is therefore not applicable to this application, and coverage of the PHP and JavaScript code was provided by Snyk, Semgrep, and ESLint instead.

7. Security Architecture and Diagrams

7.1 Authentication and Authorization Flow Diagram

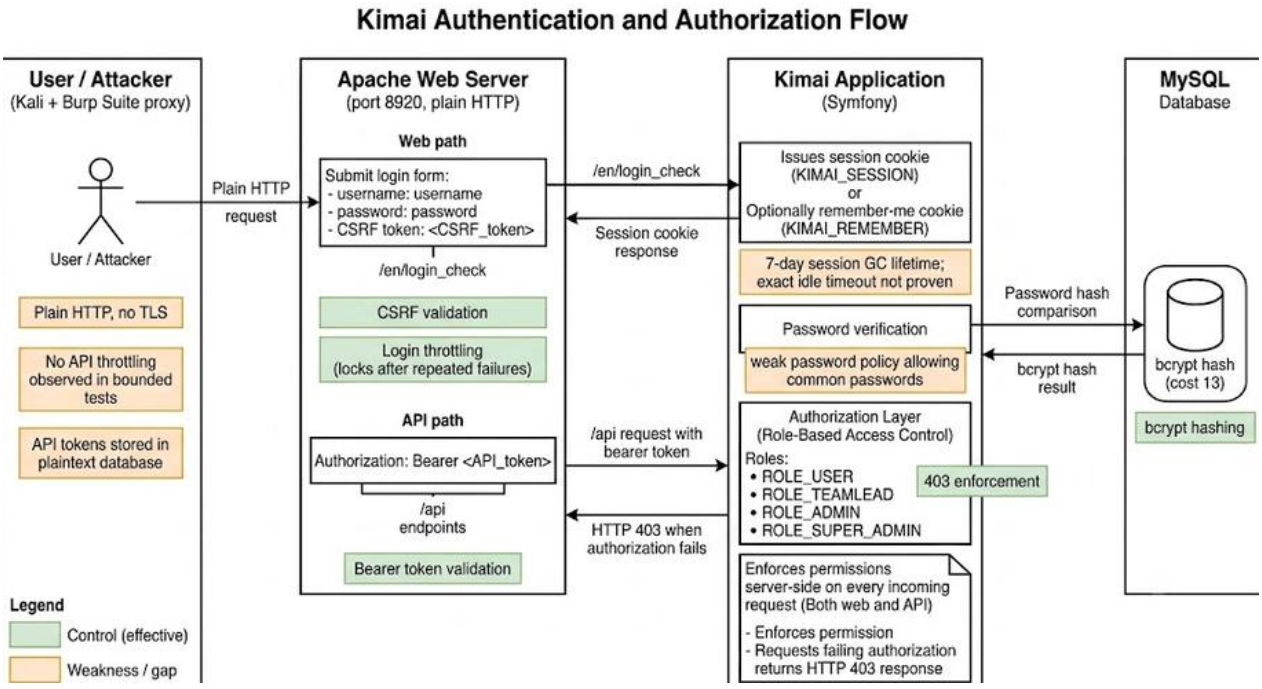


Figure 44 Authentication and authorization flow for Kimai, showing the web (session-based) and API (token-based) paths, the server-side RBAC layer, and the effective controls and identified weaknesses on each path.

Figure 44 shows how Kimai authenticates and authorizes requests across its two entry paths. On the web path, the login form submits a username, password, and CSRF token to /en/login_check, where CSRF validation and login throttling act as controls, and successful authentication issues a session cookie and optionally a remember-me cookie. On the API path, clients authenticate with a bearer API token validated on each request. Both paths converge on a server-side RBAC layer (ROLE_USER through ROLE_SUPER_ADMIN) that enforces permissions on every request and returns HTTP 403 when authorization fails. Passwords are verified against bcrypt hashes with cost 13 stored in MySQL. Effective controls are shown in green; identified weaknesses in orange.

7.2 Security Architecture Diagram

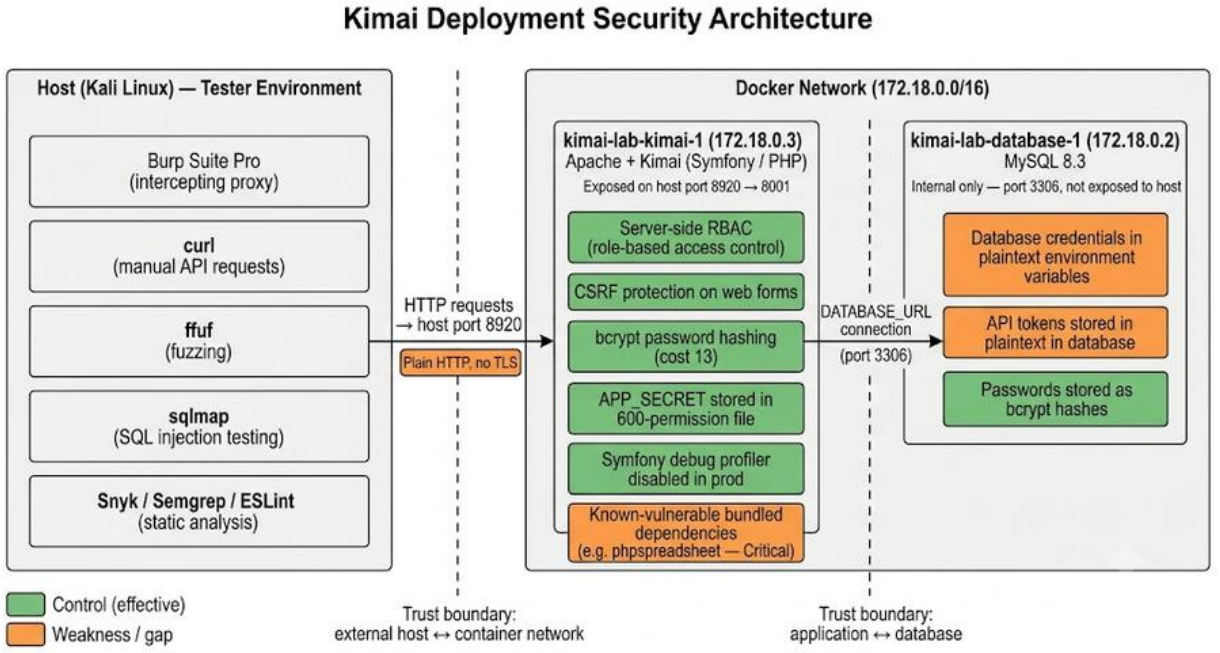


Figure 45 Security architecture of the Kimai deployment showing the tester environment, the two-container Docker network, data flows, trust boundaries, and the location of effective controls and identified weaknesses.

Figure 45 shows the deployment architecture and where each security control and weakness sits within it. The tester environment on the Kali host holds Burp Suite, curl, ffuf, sqlmap, and the static analysis tools. The application runs in two Docker containers: the Kimai application container (Apache and Symfony, 172.18.0.3, exposed on host port 8920) and the MySQL database container (172.18.0.2, internal only on port 3306). Effective controls are shown in green and sit mainly in the application container (server-side RBAC, CSRF protection, bcrypt hashing, restricted APP_SECRET file, disabled debug profiler) and the database (bcrypt password storage). Identified weaknesses are shown in orange: plain HTTP with no TLS, dependency advisories reported by Snyk with application reachability not fully confirmed, and database credentials and API tokens stored in plaintext.

8. Security Unit Testing and Automation Proposal

The testing conducted in this lab was primarily manual and tool-assisted. The following automated security tests are proposed to make coverage repeatable and to catch regressions as Kimai evolves. Examples are expressed mainly through the Kimai REST API because it exposes authentication, authorization, and validation behavior in a form that can be asserted consistently.

8.1 Input Validation Tests

A functional HTTP test should authenticate with a valid token and the required permissions, submit malformed JSON to POST /api/timesheets, and assert the expected validation status and a generic error body with no stack trace or internal path. Additional tests should change one input at a time and assert the exact status expected. Password-policy tests should assert that passwords shorter than 8 characters are rejected. A characterization test should document that 'password' is currently accepted, while the desired future security test should assert that common or compromised passwords are rejected.

8.2 Token Verification Tests

These tests verify that the API authenticates tokens correctly. They assert that a request with no Authorization header returns 401, that a request with a malformed or random bearer token returns 401, and that each token from a list of common or guessed values is rejected. A positive test asserts that a valid token authenticates successfully. A regression test for the plaintext-storage finding asserts that stored API tokens in the database are hashed; this test would currently fail, documenting the finding as a tracked, testable gap.

8.3 Role-Based Access Enforcement Tests

These tests assert that authorization boundaries hold. A standard user requesting another user's timesheet (GET, PATCH, or DELETE on /api/timesheets/{id}) should receive 403. A standard user attempting POST /api/users should receive 403. A standard user attempting to set their own roles via PATCH /api/users/{id} should be rejected because the roles field is not in their editable form. GET /api/customers should succeed (HTTP 200) while POST /api/customers returns 403, confirming the read-only-by-design behavior. Admin-only routes such as /en/admin/user/ should return 403 for a standard user session.

9. Threat and Vulnerability Coverage

9.1 STRIDE Threat Coverage (from Lab 2)

The table below maps each of the 23 STRIDE threats identified in Lab 2 to the test cases performed in Lab 3 and Lab 4, the result, and the relevant report section.

Table 3 STRIDE Threat Coverage

Threat	STRIDE	Test Case (Lab 3/4)	Result	Section
S1 Credential stuffing	Spoofing	Login throttling / weak lockout (web form)	Throttling enforced; locks after repeated failures	Auth: rate limiting
S2 API token theft	Spoofing	API token storage review + token field fuzzing	Partial: plaintext storage confirmed; stolen-token replay not tested	Secret Mgmt; Fuzzing
S3 Session fixation / hijacking	Spoofing	Lab 3 session-fixation; Lab 4 session GC review	Lab 3 covered; Lab 4 verified invalidated session rejected (exact timeout unproven)	Lab 3; Auth: session
S4 Password reset token abuse	Spoofing	Not tested	Deprioritized (mailer not configured)	n/a
T1 Timesheet modification via API	Tampering	BOLA PATCH on bob's timesheet as alice	403, blocked	API: IDOR
T2 Billing rate tampering	Tampering	Mass assignment (Lab 3)	HTTP 403 on preferences PATCH; CVE-2026-40486 not present in 2.59.0	Lab 3
T3 Invoice template injection (SSTI)	Tampering	Not re-tested in Lab 4	Covered in Lab 3; not repeated	Lab 3
T4 SQL injection	Tampering	Manual probe + sqlmap on term param	No SQL injection in tested parameter and payload set	API: injection
R1 Timesheet deletion w/o audit	Repudiation	Not tested	Deprioritized (audit-logging scope)	n/a
R2 API action without identity	Repudiation	Not tested	Deprioritized (audit-logging scope)	n/a
R3 Invoice modification denial	Repudiation	Not tested	Deprioritized (audit-logging scope)	n/a
I1 LDAP/SAML cred leak via config()	Info Disclosure	Not re-tested in Lab 4	Covered in Lab 3; not repeated	Lab 3

Threat	STRIDE	Test Case (Lab 3/4)	Result	Section
I2 APP_SECRET exposure	Info Disclosure	APP_SECRET key management review	Stored in 600-perm file; not exposed	Secret Mgmt
I3 User enumeration via login	Info Disclosure	Not tested	Deprioritized	n/a
I4 Timesheet data exposure via IDOR	Info Disclosure	BOLA GET on bob's timesheet as alice	403, blocked	API: IDOR
I5 Verbose error messages	Info Disclosure	API error handling (malformed input + fuzzing)	HTTP 400, no stack trace or debug data, no state change, no exception in logs	API: error handling
D1 Resource exhaustion via PDF	Denial of Service	Not tested	Deprioritized (stability risk)	n/a
D2 Login endpoint flooding	Denial of Service	Login throttling (web) + API rate limit test	Web form throttled; no API throttling in 30-req bounded test	Auth / API: rate limiting
D3 Bulk timesheet API abuse	Denial of Service	API rate limiting test	No throttling in 30 valid + 30 invalid req bounded test; higher volume not tested	API: rate limiting
E1 IDOR horizontal priv-esc	EoP	BOLA horizontal (alice to bob)	403, blocked	Authz: horizontal
E2 SSTI to RCE	EoP	Not re-tested in Lab 4	Covered in Lab 3; not repeated	Lab 3
E3 Stored XSS to session hijack	EoP	Not dynamically re-tested	Version 2.59.0 newer than patched; not independently confirmed	n/a
E4 Role assignment via API	EoP	Privilege escalation (alice sets own role)	HTTP 400; roles field not editable, blocked	Authz: priv-esc

Of the 23 STRIDE threats, several were directly tested, several were partially covered, and the remainder were not tested in this report. SSTI and template-injection threats (T3, I1, E2) were covered in Lab 3 and not repeated in Lab 4. The stored XSS threat (E3) was not dynamically retested; Kimai 2.59.0 is newer than the documented patched release, but that is not the same as independent confirmation. Repudiation threats (R1, R2, R3) were not tested because they require audit-log and retention analysis outside the main testing focus. The password-reset threat (S4) was not tested because the deployment's mailer was not configured. User enumeration (I3) should be tested by comparing response content and timing for existing and nonexistent users. The PDF resource-exhaustion threat (D1) was not tested because an uncontrolled test could destabilize the lab environment. Billing-rate tampering (T2) was covered in Lab 3 rather than repeated here.

9.2 Known Vulnerability Coverage (from Lab 1)

The table below maps each known vulnerability documented in Lab 1 to the test case performed in Lab 3 or Lab 4 and the result in Kimai 2.59.0.

Table 4 Known Vulnerability Coverage

Advisory	Type	Test Case (Lab 3/4)	Result in 2.59.0
GHSA-9g2q-w3w2-vf7q	Cross-team IDOR (ROLE_TEAMLEAD)	Standard-user BOLA test (alice vs. bob); team-leader scenario not reproduced	Version 2.59.0 newer than patched; cross-team scenario not independently confirmed
CVE-2023-46245	SSTI in invoice renderer	SSTI testing (Lab 3)	Tested Lab 3; sandbox blocked code execution (patched)
GHSA-jg2j-2w24-54cg	Twig sandbox bypass (SSTI)	SSTI sandbox testing (Lab 3)	Tested Lab 3; sandbox bypass not reproducible (patched)
GHSA-vrqv-52x7-rm4v	config() env/cred leak via Twig	SSTI config() probe (Lab 3)	Tested Lab 3; config() secret access blocked (patched)
CVE-2026-40479	Stored XSS via team alias	Not dynamically re-tested; version newer than patched release	Version 2.59.0 newer than patched; not independently confirmed
CVE-2026-40486	Rate field BOPA via Preferences API	Mass assignment test (Lab 3)	HTTP 403 on preferences PATCH; not present in 2.59.0
GHSA-fjhg-96cp-6fcw	Authenticated SSTI to RCE via uploaded Twig file	SSTI testing (Lab 3)	Tested Lab 3; sandbox blocked code execution (patched)
GHSA-v33r-r6h2-8wr7	Missing customer-level authorization on invoice API	Not tested (invoice endpoint; specific conditions not recreated)	Version 2.59.0 newer than patched; not independently confirmed

The standard-user BOLA tests provided related authorization evidence but did not recreate the cross-team ROLE_TEAMLEAD conditions of GHSA-9g2q-w3w2-vf7q. The SSTI and sandbox advisories were tested through the invoice template injection series in Lab 3, where the tested dangerous behaviors were not reproduced. The rate-field BOPA (CVE-2026-40486) was tested through Lab 3 mass-assignment testing and returned HTTP 403. The team-alias stored XSS condition in CVE-2026-40479 was not dynamically retested. The invoice authorization advisory GHSA-v33r-r6h2-8wr7 was not independently reproduced because its specific customer, invoice, team, and ROLE_TEAMLEAD conditions were not recreated. For dynamically tested advisories, this report states only that the tested dangerous behavior was not reproduced in this environment. For version-only comparisons, the report states that the installed version is newer than the patched release without calling that dynamic confirmation.

10. Gap Analysis and Additional Controls

This assessment confirmed several core controls in the tested Kimai 2.59.0 deployment, including server-side role-based access control in the tested standard-user scenarios, object-level authorization on the tested timesheet operations, CSRF protection on web forms, and bcrypt password hashing. No SQL injection was identified in the tested term parameter and selected payload set. The controlled malformed API input test returned a structured HTTP 400 response, exposed no stack trace, internal path, SQL error, or debug details, caused no state change, and showed no related exception in the logs for the tested case. The testing also identified a number of weaknesses and hardening observations. The following gap analysis describes the tested Docker deployment and does not assume that every setting represents Kimai's universal default configuration.

10.1 Plaintext API Token Storage at Rest (Medium - A04:2025 Cryptographic Failures)

The most significant gap is the storage of API tokens in plaintext in the database. While reading these values requires database access, storing live authentication tokens in cleartext means a single database exposure yields directly usable credentials, in contrast to user passwords, which are bcrypt-hashed. This is a design-level behavior in Kimai rather than a deployment choice. The recommended control is to store only a hash of each API token and compare hashes on authentication, so that a database compromise does not expose usable tokens.

10.2 Plain HTTP; No TLS (Medium - A02:2025 Security Misconfiguration)

The tested deployment transmits traffic over plain HTTP with no TLS. Session cookies, API tokens, and credentials are transmitted in cleartext over any network path able to reach the service. The practical severity depends on exposure: localhost-only access has less interception risk than a LAN-facing or internet-facing interface. The recommended control is to place Kimai behind an HTTPS-terminating reverse proxy with a valid certificate and to set the Secure flag on session cookies for any deployment beyond isolated local testing.

10.3 Cleartext Secrets in Environment Variables and Logs (Low - A02:2025 Security Misconfiguration)

In the tested Docker configuration, the administrative password and database connection string are present as plaintext environment variables, and the admin password is additionally written to the container logs on startup. The recommended controls are to use a secrets-management mechanism where practical, restrict access to container inspection and logs, rotate weak or exposed values, and suppress the administrative password from startup output.

10.4 Length-Only Password Policy (Low - A07:2025 Authentication Failures)

The password policy enforces a minimum length of 8 characters but does not block common passwords, so weak but length-compliant values such as 'password' are accepted. The recommended controls are to increase the minimum length, permit long passphrases, block

common or known-compromised passwords, and encourage or require two-factor authentication for sensitive accounts.

10.5 No Observed API Rate Limiting (Low - A07:2025 Authentication Failures)

No API throttling was observed during the bounded local API rate-limiting tests. Thirty valid authenticated requests returned HTTP 200 OK and thirty invalid bearer-token requests returned HTTP 401 Unauthorized without any HTTP 429 or delay increase. Broader per-IP, multi-endpoint, and higher-volume testing was not performed, so this is reported as an observation rather than a categorical absence. The recommended control is to apply a per-token and per-IP rate limiter to API endpoints, mirroring the protection already present on the login form.

10.6 Session Garbage-Collection Lifetime of 7 Days (Low - A07:2025 Authentication Failures)

PHP is configured with a session garbage-collection lifetime of 7 days, and the behavioral test confirmed that Kimai rejects an old session cookie after the matching server-side session record is removed. However, this test does not prove the exact idle timeout duration. The recommended control is to configure an explicit shorter server-side idle timeout appropriate to the application and verify rejection after real inactivity.

10.7 Outdated Dependencies with Known CVEs (Undetermined - A03:2025 Software Supply Chain)

The dependency scan identified known vulnerabilities in bundled third-party libraries, including a Critical deserialization issue in phpoffice/phpspreadsheet and medium-severity issues in mpdf and several JavaScript packages. While their exploitability within Kimai was not confirmed, outdated dependencies with known CVEs represent a supply-chain risk. The recommended control is to upgrade the flagged dependencies to their patched versions and to integrate automated dependency scanning into the development pipeline so that new advisories are detected promptly.

Appendix A: Generative AI Usage Disclosure

In accordance with the course GenAI policy, the following documents the use of generative AI tools during the preparation of this report.

Tool used: Claude (Anthropic). Claude was used to assist with the organization and professional formatting of the written report text, including rephrasing and structuring paragraphs. All test results, screenshots, tool outputs, and technical findings are based exclusively on work performed by the authors in the lab environment. No AI tool generated or fabricated any test result, screenshot, or tool output. AI-assisted text was reviewed and revised by the authors to ensure accuracy. AI-generated content constitutes less than 25% of the total text in this report.

All prompts and AI-generated content are available upon request.

Appendix B: References

1. OWASP Foundation. OWASP Web Security Testing Guide v4.2. https://owasp.org/www-project-web-security-testing-guide/

2. OWASP Foundation. OWASP Top 10 2025. https://owasp.org/www-project-top-ten/

3. Kimai Project. Kimai Time Tracking v2.59.0. https://github.com/kimai/kimai

4. Snyk Ltd. Snyk Open Source Security Platform. https://snyk.io

5. Semgrep Inc. Semgrep Static Analysis Tool. https://semgrep.dev

6. PortSwigger Ltd. Burp Suite Community Edition. https://portswigger.net/burp

7. ffuf Project. Fast Web Fuzzer (ffuf). https://github.com/ffuf/ffuf

8. sqlmap Project. Automatic SQL Injection Tool. https://sqlmap.org

9. Daniel Miessler. SecLists. https://github.com/danielmiessler/SecLists