Web Application & API Security Assessment

Prepared by	Pentest Testing Corp | https://www.pentesttesting.com/
Targets	https://example.com (Web App) · https://ptest.example.com.au (SaaS/API)
Report Date	June 08, 2026
Classification	CONFIDENTIAL – Client Use Only
Version	1.0 – Final
Overall Risk	HIGH

Finding Severity Summary

CRITICAL	HIGH	MEDIUM	LOW	INFO
1	7	3	3	1

This report is prepared exclusively for the named client organisation. It contains sensitive security findings and must not be distributed beyond authorised personnel. All findings, evidence and reproduction steps have been sanitised to remove live credentials, PII, and customer data.

Table of Contents

1. **Executive Summary** 3
2. **Scope & Methodology** 4
3. **Consolidated Findings Summary** 5
4. **Detailed Findings — Web Application** 6
   - 4.1 SQL Injection (F-WA-01) 6
   - 4.2 Cross-Site Scripting / XSS (F-WA-02) 8
   - 4.3 Broken Authentication & Session Fixation (F-WA-03) 9
   - 4.4 Sensitive Data Exposure (F-WA-04) 10
   - 4.5 Cross-Site Request Forgery (F-WA-05) 10
   - 4.6 Security Misconfiguration (F-WA-06) 11
   - 4.7 Malicious File Upload (F-WA-07) 12
   - 4.8 Known Vulnerable Components (F-WA-08) 13
   - 4.9 Banner / Information Disclosure (F-WA-09) 13
5. **Detailed Findings — SaaS / API** 14
   - 5.1 BOLA/IDOR – Job Posts (F-API-01) 14
   - 5.2 BOLA – Chat Cross-Account Access (F-API-02) 15
   - 5.3 Broken Function-Level Authorization (F-API-03) 16
   - 5.4 Session Management – Token Non-Invalidation (F-API-04) 17
   - 5.5 Improper Rate Limiting (F-API-05) 17
   - 5.6 User Enumeration (F-API-06) 18
   - 5.7 Verbose SQL Error Disclosure (F-API-07) 18
   - 5.8 Mass Assignment / Overposting (F-API-08) 19
   - 5.9 Pagination Abuse (F-API-09) 19
   - 5.10 Missing Security Headers & Permissive CORS (F-API-10) 20
   - 5.11 S3 Bucket CORS Misconfiguration (F-API-11) 20
6. **CVSS & OWASP Mapping Table** 21

7.	Remediation Roadmap	22
8.	Retest Status	23
9.	Appendices	24

1. Executive Summary

Pentest Testing Corp conducted a comprehensive black-box penetration test against two target environments: a legacy PHP municipal web application and a modern SaaS pharmacy staffing API platform. Testing covered authentication, session management, authorisation (RBAC and object-level), API security, input validation, and server configuration.

Key Findings at a Glance

- **Critical (1):** Broken Function-Level Authorisation permits any authenticated user to access group-manager financial data (invoices, services).
- **Critical (1, combined BOLA):** Cross-tenant job-post read/write/create enables full multi-tenant data isolation breach.
- **High (7):** SQL Injection, XSS (reflected & stored), Sensitive Data Exposure, CSRF, Malicious File Upload, Verbose SQL Error Disclosure, Token Non-Invalidation on Logout.
- **Medium (3):** Improper Rate Limiting (brute-force), User Enumeration, Pagination Abuse.
- **Low (3):** Banner Disclosure, Password Autocomplete, Missing Security Headers / Permissive CORS / S3 CORS.
- **Informational (1):** Undocumented endpoint discovery.

SOC 2 Relevance

Multiple findings directly affect SOC 2 Trust Service Criteria. The authorization failures (F-API-01, F-API-03) violate CC6.1 (Logical Access) and CC6.3 (Access Restrictions). Token non-invalidation (F-API-04) violates CC6.1 session controls. SQL Injection and CSRF (F-WA-01, F-WA-05) violate CC7.1 (Vulnerability Management). Verbose error disclosure and misconfiguration issues affect CC7.2 (Monitoring). All critical and high findings should be remediated before the next SOC 2 audit cycle.

Overall Risk Rating: HIGH

The combination of tenant isolation failures, unauthenticated data exposure, and server-side injection vulnerabilities represents a high aggregate risk. Immediate remediation of critical and high findings is strongly recommended.

2. Scope & Methodology

2.1 Scope

Asset	URL / Target	Type
Web Application	https://example.com/	PHP Municipal App
SaaS API (Staging)	https://ptest.example.com.au	React + REST API
SaaS API Backend	https://ptestback.example.com.au/api/v1	Laravel API
Cloud Storage	example-staging.s3.ap-southeast-2.amazonaws.com	AWS S3

2.2 Accounts / Roles Used

- Pharmacy Account A
- Pharmacy Account B
- Locum Account
- Job Seeker Account
- Admin / Superadmin (web app)

2.3 Methodology

Severity	CVSS Range	Description
Critical	9.0–10.0	Immediate exploitation risk; full system/tenant compromise possible.
High	7.0–8.9	Significant data exposure or application compromise likely.
Medium	4.0–6.9	Exploitation requires additional conditions; moderate impact.
Low	0.1–3.9	Minimal direct impact; aids attacker reconnaissance.
Info	0.0	Informational observations with no immediate exploitability.

Reconnaissance	Passive and active enumeration of endpoints, headers, technologies, and error messages.
Authentication Testing	Credential brute-force resistance, session fixation, token lifecycle, logout invalidation.
Authorisation Testing	RBAC and BOLA/IDOR testing by replaying requests with tokens from different roles and tenants; object identifier manipulation.
Input Validation	Manual injection testing (SQL, XSS, CSRF, file upload) against all identified input parameters.
API-Specific	OWASP API Security Top 10 checklist: BOLA, BFLA, mass assignment, rate limiting, user enumeration, verbose errors.
Infrastructure	HTTP method probing, header analysis, CORS, S3 bucket configuration review.
Tools Used	Burp Suite Professional, cURL scripting, ffuf (endpoint fuzzing), OWASP ZAP, sqlmap (validation), manual review.

2.4 Severity Rating Scale

3. Consolidated Findings Summary

ID	Title	Severity	CVSS	OWASP	Status
F-WA-01	SQL Injection	High	8.8	A1 / API1	Open
F-WA-02	Cross-Site Scripting (Reflected & Stored)	High	8.2	A7	Open
F-WA-03	Broken Auth & Session Fixation	High	7.5	A2	Open
F-WA-04	Sensitive Data Exposure (Plaintext Creds)	High	7.4	A3	Open
F-WA-05	Cross-Site Request Forgery	High	8.0	A8 (2013)	Open
F-WA-06	Security Misconfiguration	High	7.2	A6 / API8	Open
F-WA-07	Malicious File Upload / RCE	High	9.0	A1	Open
F-WA-08	Known Vulnerable Components	Medium	6.3	A9	Open
F-WA-09	Banner & Info Disclosure	Low	3.1	A6	Open
F-API-01	BOLA/IDOR – Job Posts (Cross-Tenant)	Critical	9.1	API1:2023	Open
F-API-02	BOLA – Chat Cross-Account Access	High	8.1	API1:2023	Open
F-API-03	Broken Function-Level Authorization	Critical	9.3	API5:2023	Open
F-API-04	Token Non-Invalidation on Logout	High	7.6	API2:2023	Open
F-API-05	Improper Rate Limiting	Medium	5.3	API4:2023	Open
F-API-06	User Enumeration	Medium	5.0	API3:2023	Open
F-API-07	Verbose SQL Error Disclosure	High	7.5	API8:2023	Open
F-API-08	Mass Assignment / Overposting	High	8.0	API3:2023	Open
F-API-09	Pagination Abuse & Insufficient Rate Limiting	Medium	4.3	API4:2023	Open
F-API-10	Missing Security Headers & Permissive CORS	Low	3.5	API8:2023	Open
F-API-11	S3 CORS Misconfiguration	Low	3.0	API8:2023	Open
F-API-12	Undocumented Endpoint Discovery	Info	0.0	API9:2023	Open

4. Detailed Findings — Web Application

The following findings were identified during manual and automated testing of the PHP-based municipal web application at https://example.com/.

F-WA-01 – SQL Injection	High
CVSS Score	8.8 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)
OWASP Mapping	OWASP A1:2017 – Injection | OWASP API1:2023 BOLA (secondary) | CWE-89
Affected	https://example.com/get_designations.php, /manage_comp.php, /ajax_getdists2.php (+36 endpoints)
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC7.1 (Vulnerability Management), CC6.6 (Logical Access)

Description

Multiple GET and POST parameters are vulnerable to SQL Injection due to the lack of parameterised queries. An unauthenticated or low-privileged attacker can extract the full database schema and contents, modify records, and — given the overprivileged DB connection — issue operating system commands. The backend DBMS was fingerprinted as MySQL/MariaDB 10.0.38 running on Apache/PHP 5.6.40.

Evidence / Proof of Concept

Single-quote injection in the dept_id parameter of get_designations.php returned a raw MySQL error exposing the query structure. Automated extraction enumerated 142 tables in the municipa_csms database including user credentials, complaint records, and administrative data. The PHP backdoor uploaded via the file-upload vulnerability independently confirmed OS-level access.

Steps to Reproduce (Sanitized)

- Navigate to: GET /get_designations.php?dept_id=24'
- Observe MySQL SQLSTATE error message in response body.
- Run: sqlmap -u 'http://example.com/get_designations.php?dept_id=24' --dbs
- Confirm database enumeration returns municipa_csms with 142 tables.
- Verify similarly on /ajax_getulbs2.php?distid=1' and /manage_comp.php POST body.

Remediation Guidance

- Replace all dynamic SQL with parameterised queries or prepared statements (PDO / MySQLi).
- Apply strict server-side whitelist input validation (type-check, length-check) on all parameters.
- Use escaping routines only as a secondary defence — not a primary fix.
- Suppress all database error output to clients; log errors server-side with a correlation ID.
- Reduce DB user privileges to the minimum required (SELECT/INSERT only where possible).
- Deploy a Web Application Firewall (WAF) as a compensating control.

F-WA-02 – Cross-Site Scripting (Reflected & Stored)	High
CVSS Score	8.2 (AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N)

OWASP Mapping	OWASP A7:2017 – XSS | CWE-79
Affected	https://example.com/cdma_ulbwise_report.php, /videos (admin), /update_emp.php (+12 endpoints)
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.6, CC7.1

Description

The application reflects unsanitised user input directly into HTML responses (reflected XSS) and persists malicious payloads in the database (stored XSS). Both variants allow an attacker to execute arbitrary JavaScript in a victim's browser, enabling session hijacking, credential harvesting, defacement, and redirection to malicious sites. 150 reflected XSS instances and multiple stored XSS vectors were identified.

Evidence / Proof of Concept

Injecting the payload alert(document.domain) via the app_type_id parameter of cdma_ulbwise_report.php returned the payload unencoded in the HTML response and triggered execution. A stored XSS via the video-update admin form persisted the payload and executed it for every user visiting the Videos admin page.

Steps to Reproduce (Sanitized)

- GET /cdma_ulbwise_report.php?status=0&app;_type_id=1">alert(1)
- Observe script execution in browser (reflected).
- In admin panel, submit video update form with body field containing alert('xss').
- Navigate to Videos admin page — observe payload fires for every page load (stored).

Remediation Guidance

- Perform context-sensitive output encoding on all user-supplied data before rendering (HTML entity encoding,JavaScript encoding, URL encoding as appropriate).
- Implement a Content Security Policy (CSP) header with a restrictive default-src directive.
- Use a framework-level encoding library (e.g., OWASP Java Encoder, AntiXSS) applied consistently.
- Validate and sanitise input server-side — disallow or strip dangerous HTML tags in stored fields.
- Set the HttpOnly and Secure flags on all session cookies.

F-WA-03 – Broken Authentication & Session Fixation — High

CVSS Score	7.5 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)
OWASP Mapping	OWASP A2:2017 – Broken Authentication | CWE-384
Affected	https://example.com/check_login.php (POST)
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.1, CC6.2

Description

The application retains the same PHP session ID (PHPSESSID) before and after login, enabling session fixation attacks. An attacker who can deliver a known PHPSESSID to a victim (via URL injection, XSS, or subdomain takeover) can hijack the authenticated session without knowing the user's credentials. Additionally, no old-password verification is required when changing passwords, allowing account takeover via CSRF.

Evidence / Proof of Concept

Burp Suite capture confirmed that the PHPSESSID cookie value in the login POST request was identical to the value returned in the authenticated dashboard GET response. The password-change page for both admin and ULB roles accepted a new password with only a blank 'old password' field.

Steps to Reproduce (Sanitized)

- Capture PHPSESSID prior to login.
- Complete login; observe the same PHPSESSID is maintained post-authentication.
- From a second browser with that PHPSESSID, access /dashboard.php — session is active.
- Navigate to /change_pwd.php; submit new password without providing current password — succeeds.

Remediation Guidance

- Regenerate the session ID immediately after successful authentication using session_regenerate_id(true) in PHP.
- Destroy and regenerate the session on logout.
- Require current password verification on all password-change operations.
- Apply Secure, HttpOnly, and SameSite=Strict flags to the session cookie.
- Implement account lockout after repeated failed authentication attempts.

F-WA-04 – Sensitive Data Exposure — Plaintext Credentials — High

CVSS Score	7.4 (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N)
OWASP Mapping	OWASP A3:2017 – Sensitive Data Exposure | CWE-311, CWE-319
Affected	https://example.com/ (entire application, transmitted over HTTP)
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.7 (Encryption in Transit), CC9.2

Description

The application transmits user credentials and session tokens over plain HTTP without TLS encryption. Network-level interception (on-path attackers, malicious Wi-Fi) can expose passwords, session cookies, and PII in cleartext. The login POST request and subsequent session traffic were observed without HTTPS in Wireshark captures.

Evidence / Proof of Concept

Wireshark frame capture showed the login POST body containing username, password, and captcha in cleartext on the wire. The password hash and salt were also returned in the authentication response body, exposing them to anyone with network access.

Steps to Reproduce (Sanitized)

- Start a network capture on the same network segment.
- Submit login credentials on http://example.com/check_login.php.
- Filter Wireshark for HTTP POST — observe plaintext credentials in frame payload.

Remediation Guidance

- Deploy a trusted TLS certificate and enforce HTTPS across all application routes.
- Implement HSTS (Strict-Transport-Security) with a minimum max-age of 1 year.
- Never return password hashes or salts to the client.
- Mark all session cookies as Secure so they are never transmitted over plain HTTP.

F-WA-05 – Cross-Site Request Forgery (CSRF) High

CVSS Score	8.0 (AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N)
OWASP Mapping	OWASP A8:2013 – CSRF | CWE-352
Affected	All stateful forms at https://example.com/ and https://example.com/sites/
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.6, CC6.8

Description

All sensitive state-changing operations (user management, designation updates, employee mapping) lack CSRF tokens. An attacker can craft a forged HTML page that, when visited by an authenticated user, silently performs actions (e.g., modify user data, change designations) on behalf of that user. 160 CSRF-vulnerable instances were identified.

Evidence / Proof of Concept

A proof-of-concept HTML form was crafted that auto-submits a POST to /update_emp.php with forged employee details. When a logged-in administrator clicked the attacker-sent link, the employee record was updated without any CSRF challenge.

Steps to Reproduce (Sanitized)

- Capture a legitimate POST to /update_emp.php using Burp.
- Generate CSRF PoC HTML (Burp: Engagement tools > Generate CSRF PoC).
- Host the HTML page and trick an authenticated admin into visiting it.
- Observe the employee record is updated without the admin's awareness.

Remediation Guidance

- Implement synchronised CSRF tokens on all state-changing forms and validate server-side.
- Use the SameSite=Strict cookie attribute to prevent cross-origin cookie submission.
- Require re-authentication for high-impact operations (user deletion, privilege changes).
- Validate the Origin and Referer headers as a secondary defence.

F-WA-06 – Security Misconfiguration High

CVSS Score	7.2 (AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N)
OWASP Mapping	OWASP A6:2017 – Security Misconfiguration | OWASP API8:2023
Affected	https://example.com/ (server-wide)
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC7.1, CC6.6

Description

Multiple misconfiguration issues were identified: (1) PHP warnings expose physical file paths and SQL queries in page output; (2) phpinfo() is publicly accessible, disclosing full server configuration; (3) Old password validation is absent in password-change flows; (4) The HTTPOnly flag is missing on session cookies; (5) Source code fragments appear in Burp response bodies.

Evidence / Proof of Concept

Accessing /info.php returned PHP 5.6.40 server configuration including build flags, environment variables, and extension details. PHP warning messages in page headers disclosed paths such as /home2/municipalservce/public_html/connection.php.

Steps to Reproduce (Sanitized)

- Navigate to http://example.com/info.php — full phpinfo() output disclosed.
- Trigger any form validation error — PHP warning discloses server file path in response.
- Inspect cookie in browser DevTools — HTTPOnly flag absent on PHPSESSID.

Remediation Guidance

- Remove or restrict access to phpinfo() and any debug/diagnostic pages in production.
- Set display_errors = Off and log_errors = On in php.ini; suppress all runtime errors from client output.
- Set expose_php = Off to suppress the X-Powered-By: PHP header.
- Enable the HttpOnly and Secure flags on session cookies.
- Remove backup files, source fragments, and temporary files from web-accessible directories.
- Apply timely security patches for PHP, Apache, and all third-party dependencies.

F-WA-07 – Malicious File Upload / Remote Code Execution — High

CVSS Score	9.0 (AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H)
OWASP Mapping	OWASP A1:2017 – Injection (unrestricted upload) | CWE-434
Affected	https://example.com/add_public_rep.php and all file upload endpoints
Retest Status	Open – URGENT (Active Backdoor Present)
SOC 2 Criteria	CC7.2, CC6.8

Description

The application accepts file uploads without validating file extension, MIME type, or content. A PHP webshell was successfully uploaded and executed via the photo upload functionality, providing full server-side code execution. The shell permitted directory traversal, MySQL query execution, OS command execution, and retrieval of /etc/passwd.

Evidence / Proof of Concept

A PHP backdoor file (info.php) was uploaded by selecting it in the 'Photo' field of the Add Public Representatives form. The server returned 'Saved Successfully' and the file was accessible at the predictable path /council/0/15514227260.php, providing a web-accessible shell with OS-level privileges.

Steps to Reproduce (Sanitized)

- Navigate to /add_public_rep.php and submit the form with a .php file in the Photo field.
- Observe 'Saved Successfully' response — no file type rejection.
- Access the uploaded file URL directly — web shell interface loads.
- Execute OS command: cat /etc/passwd — output returned in browser.

Remediation Guidance

- Implement a server-side whitelist of permitted file extensions (e.g., jpg, png, gif, pdf).
- Validate file content using MIME type detection (not only the Content-Type header).
- Store uploaded files outside the web root and serve them through a dedicated controller that streams content.
- Randomise uploaded filenames to prevent direct access to known paths.

- Scan uploaded files with an antivirus/antimalware engine before storing.
- **URGENT: Remove the existing PHP backdoor from /council/0/15514227260.php immediately.**

F-WA-08 – Known Vulnerable Components — Medium

CVSS Score	6.3 (AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N)
OWASP Mapping	OWASP A9:2017 – Using Components with Known Vulnerabilities | CWE-1104
Affected	https://example.com/ (jQuery UI 1.10.3, Bootstrap 3.3.6, CKEditor 4.4.2)
Retest Status	Open – Pending Fix

Description

The application uses multiple client-side libraries with publicly known CVEs. jQuery UI 1.10.3 (EOL) and Bootstrap 3.3.6 contain XSS vulnerabilities. CKEditor 4.4.2 has multiple documented security issues. Exploitation of these components can assist in escalating other vulnerabilities.

Evidence / Proof of Concept

Browser inspection of /js/jquery-ui-1.10.3.custom.min.js and /js/bootstrap.min.js confirmed version strings. The NIST NVD lists multiple CVEs against these versions including CVE-2016-7103 (jQuery UI XSS), CVE-2019-8331 (Bootstrap XSS).

Steps to Reproduce (Sanitized)

- View page source or inspect network tab — version strings visible in library URLs/comments.
- Cross-reference versions against NIST NVD or Snyk vulnerability database.

Remediation Guidance

- Upgrade jQuery UI to the latest stable version (3.x+).
- Upgrade Bootstrap to the latest stable version (5.x).
- Upgrade CKEditor to the latest version (5.x) or consider migration to a maintained alternative.
- Implement Software Composition Analysis (SCA) tooling in the CI pipeline (e.g., OWASP Dependency-Check,Snyk).

F-WA-09 – Banner & Information Disclosure — Low

CVSS Score	3.1 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP A6:2017 – Security Misconfiguration | CWE-200
Affected	https://example.com/ (all responses)
Retest Status	Open – Pending Fix

Description

HTTP responses include the X-Powered-By: PHP/5.6.40 header and Server: Apache/2.4 header, disclosing exact technology versions to attackers. This facilitates targeted exploitation using version-specific vulnerabilities. Both PHP 5.6 and the detected Apache version are end-of-life with no further security support.

Evidence / Proof of Concept

Burp Suite response inspection confirmed X-Powered-By and Server headers on all application responses. PHP 5.6 reached end of life in December 2018.

Steps to Reproduce (Sanitized)

- Send any HTTP request and inspect response headers in Burp or browser DevTools.
- Observe X-Powered-By: PHP/5.6.40 and Server: Apache in response.

Remediation Guidance

- Set expose_php = Off in php.ini to suppress the X-Powered-By header.
- Configure ServerTokens Prod and ServerSignature Off in Apache httpd.conf.
- Upgrade PHP to a supported version (8.2+) and upgrade Apache.
- Suppress Server header at the reverse proxy/load balancer layer.

5. Detailed Findings — SaaS / API

The following findings were identified during testing of the SaaS pharmacy staffing platform at https://ptest.example.com.au and its backend API at https://ptestback.example.com.au/api/v1.

F-API-01 – BOLA/IDOR – Job Posts Cross-Tenant Access & Modification	Critical
CVSS Score	9.1 (AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N)
OWASP Mapping	OWASP API1:2023 – Broken Object Level Authorization | CWE-639
Affected	GET/PUT /api/v1/job-board/job-post/{job_post_id} | POST /api/v1/job-board/job-post
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.1 (Logical Access Controls), CC6.3 (Authorization)

Description

The API performs no server-side ownership verification when a user accesses or modifies a job post. Any authenticated user can read, update, or create job posts belonging to a different pharmacy tenant by supplying an arbitrary job_post_id or pharmacy_id in the request. This constitutes a full multi-tenant data isolation failure and violates the fundamental security boundary of the SaaS platform.

Evidence / Proof of Concept

Using Pharmacy User B's bearer token, a GET request to /api/v1/job-board/job-post/{User_A_job_post_id} returned 200 OK with full job details. A PUT to the same endpoint with a modified job_title returned 'Job updated successfully'. A POST to /job-board/job-post with pharmacy_id set to User A's identifier created a job post attributed to User A's account.

Steps to Reproduce (Sanitized)

- Login as Pharmacy User B; capture bearer token from /api/v1/login.
- Identify a valid job_post_id belonging to Pharmacy User A.
- GET /api/v1/job-board/job-post/{UserA_ID} with User B token → 200 OK (should be 403).
- PUT /api/v1/job-board/job-post/{UserA_ID} with {"job_title":"Hacked"} → 200 OK.
- POST /api/v1/job-board/job-post with pharmacy_id = User A's ID → job created under User A.

Remediation Guidance

- Enforce server-side ownership checks: resolve the authenticated user's tenant/pharmacy from the access token andreject any request where job_post_id does not belong to that tenant.

- Never accept pharmacy_id from the client — derive it exclusively from the authenticated token subject.
- Implement a consistent authorization middleware applied to all job-board controllers.
- Add integration tests that verify cross-tenant access returns 403 for every endpoint.
- Enable audit logging for all cross-tenant access attempts with alerting on authorization failures.

F-API-02 – BOLA – Chat / Conversation Cross-Account Access — High

CVSS Score	8.1 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N)
OWASP Mapping	OWASP API1:2023 – BOLA | CWE-639
Affected	GET /api/v1/chat/{chat_id}
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.1, CC6.3

Description

The chat endpoint does not validate that the requesting user is a participant in the conversation referenced by chat_id. Any authenticated user (regardless of role or tenant) can retrieve full message histories, sender/receiver identifiers, and timestamps for any chat by supplying an arbitrary chat_id.

Evidence / Proof of Concept

Using User B's bearer token, a GET request to /api/v1/chat/{User_A_chat_id} returned 200 OK with the full message array including content, sender_id, receiver_id, and timestamps belonging to User A's conversation.

Steps to Reproduce (Sanitized)

- Authenticate as User A; capture a chat_id from a network request.
- Authenticate as User B; capture bearer token.
- GET /api/v1/chat/{UserA_chat_id} with User B token → 200 OK with User A messages (should be 403).

Remediation Guidance

- Validate on every request that the authenticated principal is a participant in the referenced chat (membership checkbefore returning any data).
- Use non-guessable chat identifiers (UUIDs) as a supplementary control — not a replacement for authorization.
- Apply per-tenant database query scoping: WHERE tenant_id = current_tenant on all chat queries.

F-API-03 – Broken Function-Level Authorization — Group Manager Endpoints — Critical

CVSS Score	9.3 (AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N)
OWASP Mapping	OWASP API5:2023 – Broken Function Level Authorization | CWE-285
Affected	GET /api/v1/group-manager/services | GET /api/v1/group-manager/invoices
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.1, CC6.3, CC9.2

Description

Endpoints designated for the group-manager role return sensitive financial data (invoice IDs, dates, amounts, customer names) to any authenticated user regardless of role. Non-group-manager tokens (pharmacy, locum, job seeker)

received 200 OK responses with invoice listings. Additional group-manager endpoints return 500 Internal Server Error instead of 403, leaking internal error details.

Evidence / Proof of Concept

Pharmacy, Locum, and Job Seeker bearer tokens all returned 200 OK on GET /api/v1/group-manager/invoices, exposing invoice listings with financial data. Requests to /api/v1/group-manager/dashboard and /approved-timesheets returned 500 errors with stack trace fragments rather than 403 Forbidden.

Steps to Reproduce (Sanitized)

- Authenticate as Pharmacy or Job Seeker role; capture bearer token.
- GET /api/v1/group-manager/invoices with that token → 200 OK + invoice data (should be 403).
- GET /api/v1/group-manager/services → 200 OK (should be 403).
- GET /api/v1/group-manager/dashboard → 500 Internal Server Error (should be 403).

Remediation Guidance

- Implement explicit server-side RBAC middleware for every group-manager route — verify role claim in token beforeprocessing request.
- Apply a deny-by-default policy: unrecognised or insufficient roles receive 403 Forbidden.
- Return consistent 403/401 responses; never expose stack traces or internal error details to clients.
- Add automated authorization tests in CI: for each role × endpoint combination, verify expected status codes.

F-API-04 – Session Management — Token Non-Invalidation on Logout

High

CVSS Score	7.6 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)
OWASP Mapping	OWASP API2:2023 – Broken Authentication | CWE-613
Affected	Bearer tokens (all API endpoints); Web logout flow
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.1, CC6.2

Description

Bearer tokens remain valid after the user initiates logout via the web UI. Tokens issued earlier in the session also continue to function, indicating no server-side revocation mechanism exists. A stolen token (via XSS, log exposure, or endpoint compromise) grants persistent access with no forced expiry pathway.

Evidence / Proof of Concept

After completing the web UI logout flow, previously captured bearer tokens returned 200 OK responses on protected endpoints including /api/v1/me and /api/v1/job-board/job-post. Token age had no observable effect on validity.

Steps to Reproduce (Sanitized)

- Login and capture Authorization: Bearer from API requests.
- Initiate logout via the web UI.
- Replay captured requests with old bearer token → 200 OK (should be 401).

Remediation Guidance

- Implement server-side token revocation: maintain a token blocklist (Redis/DB) or use token versioning; invalidate onlogout, password change, and account deactivation.

- Use short-lived access tokens (15 minutes) paired with revocable refresh tokens.
- On logout, revoke both access and refresh tokens server-side.
- Bind tokens to device/session identifiers where feasible; rotate tokens on sensitive account changes.
- Store tokens in HttpOnly Secure cookies rather than localStorage to mitigate XSS theft.

F-API-05 – Improper Rate Limiting — No Effective Lockout — Medium

CVSS Score	5.3 (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP API4:2023 – Unrestricted Resource Consumption | CWE-307
Affected	POST /api/v1/login | Listing endpoints
Retest Status	Open – Pending Fix

Description

High-volume authentication attempts (>2,500 requests) did not trigger a 429 Too Many Requests response or account lockout. Rate-limit headers (X-RateLimit-*) were present but appeared static and were not enforced. This exposes the platform to credential stuffing and brute-force attacks.

Evidence / Proof of Concept

A scripted loop of 2,500 invalid login requests sustained consistent 400 responses with no throttling, lockout, or 429 observed. X-RateLimit-Remaining decreased but never triggered a blocking response.

Steps to Reproduce (Sanitized)

- Script: for i in $(seq 1 2500); do curl -s -o /dev/null -w "%{http_code}" -X POST BASE/api/v1/login -d'email=test&password;=wrong'; done
- Observe all responses are 400 — no 429 or lockout triggered.

Remediation Guidance

- Implement rate limiting keyed on IP + username with exponential backoff: e.g., lock for 5 minutes after 10 failures.
- Return 429 Too Many Requests with a Retry-After header when limits are exceeded.
- Add CAPTCHA or step-up verification after repeated failures.
- Deploy IP-based anomaly detection and alerting for abnormal authentication traffic volumes.

F-API-06 – User Enumeration via /user-exists — Medium

CVSS Score	5.0 (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP API3:2023 – Broken Object Property Level Authorization | CWE-203
Affected	GET /api/v1/user-exists?login={email}
Retest Status	Open – Pending Fix

Description

The endpoint returns an explicit boolean indicating whether an account exists for a given login value. This enables attackers to enumerate valid email addresses at scale, dramatically increasing the efficiency of phishing campaigns and credential-stuffing attacks.

Evidence / Proof of Concept

Requests with registered emails returned {"status":true}; unregistered emails returned {"status":false}. No rate limiting was applied to this endpoint.

Steps to Reproduce (Sanitized)

- GET /api/v1/user-exists?login=valid@example.com → {"status":true} • GET /api/v1/user-exists?login=notexist@example.com → {"status":false}
- Script enumeration using a wordlist of email addresses.

Remediation Guidance

- Return a generic neutral response regardless of account existence (e.g., always 200 with a non-committalmessage).
- Apply aggressive rate limiting and CAPTCHA challenges for unauthenticated calls.
- Log and alert on bulk enumeration patterns from single IPs.

F-API-07 – Verbose SQL / Database Error Disclosure — High

CVSS Score	7.5 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N)
OWASP Mapping	OWASP API8:2023 – Security Misconfiguration | CWE-209
Affected	POST /api/v1/job-board/job-post
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC7.2

Description

500 Internal Server Error responses include full SQLSTATE error messages, table and column names (e.g., platform_role_id), and partial query structures. This directly aids attackers in crafting targeted SQL injection payloads and understanding the internal data model without authentication escalation.

Evidence / Proof of Concept

Submitting malformed input (wrong datatype for platform_role_id) to the job-post creation endpoint returned a 500 response containing: SQLSTATE[23000]: Integrity constraint violation… INSERT INTO `job_posts` (`platform_role_id`, `pharmacy_id`…) with table and column names fully visible.

Steps to Reproduce (Sanitized)

- POST /api/v1/job-board/job-post with platform_role_id set to a string value.
- Observe 500 response body containing SQLSTATE error with table/column names.

Remediation Guidance

- Disable verbose exception output in staging/production; return generic error messages with a correlation ID.
- Implement centralised error handling middleware that catches all exceptions and normalises responses.
- Log full stack traces server-side only, accessible by operations staff via secure log aggregation.
- Ensure ORM/query builder errors are caught before reaching the HTTP response layer.

F-API-08 – Mass Assignment / Overposting in Job Post Creation — High

CVSS Score	8.0 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N)

OWASP Mapping	OWASP API3:2023 – Broken Object Property Level Authorization | CWE-915
Affected	POST /api/v1/job-board/job-post
Retest Status	Open – Pending Fix
SOC 2 Criteria	CC6.3

Description

The job post creation endpoint accepts client-supplied parameters that appear to be server-controlled, including role/permission flags, approval fields, and platform_role_id. The server returns 200 OK and creates the resource, indicating insufficient server-side field allowlisting. This may enable privilege escalation, unauthorised publication, or workflow bypass.

Evidence / Proof of Concept

A POST request with additional undocumented body parameters (including is_admin, approval_status, platform_role_id) returned 200 OK and the job post was created with the attacker-supplied values persisted in the response.

Steps to Reproduce (Sanitized)

- Capture a standard POST /api/v1/job-board/job-post request.
- Add additional body fields: is_admin=true, approval_status=approved, platform_role_id=1.
- Observe 200 OK — server accepts and persists undocumented fields.

Remediation Guidance

- Implement strict server-side allowlists (Data Transfer Objects / DTOs) for all create and update operations —unknown fields must be rejected.
- Enforce server-side authorization on sensitive fields (status, approval, role IDs, group_id) — these must never beclient-settable.
- Add negative tests in CI/CD for overposting: confirm restricted fields return 400 or are silently ignored with nopersistence.

F-API-09 – Pagination Abuse & Insufficient Rate Limiting on Listing Endpoints — Medium

CVSS Score	4.3 (AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP API4:2023 – Unrestricted Resource Consumption | CWE-400
Affected	GET /api/v1/job-board/job-post | Listing endpoints
Retest Status	Open – Pending Fix

Description

The API accepts arbitrarily large pagination values (per_page up to 5000+) and sustained high-volume request loops without effective throttling. This enables bulk data harvesting at scale and increases backend load, creating a risk of service degradation.

Evidence / Proof of Concept

Requests with per_page=5000 returned 200 OK with significantly larger response bodies. A loop of 1,200 sequential requests returned sustained 200 responses with no observable throttling beyond static X-RateLimit header values.

Steps to Reproduce (Sanitized)

- GET /api/v1/job-board/job-post?per_page=5000 → 200 OK, large payload.

• Script 1,200 sequential requests — no 429 observed.

Remediation Guidance

• Enforce a maximum page size server-side (e.g., 50–200 records); return 400 for out-of-range values.
• Implement robust per-user/IP/token rate limiting with 429 + Retry-After enforcement.
• Add cursor-based pagination to prevent offset-based enumeration of large datasets.

F-API-10 – Missing Security Headers & Permissive CORS — Low

CVSS Score	3.5 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP API8:2023 – Security Misconfiguration | CWE-16
Affected	Web + API responses (sampled on / and /api/v1/me)
Retest Status	Open – Pending Fix

Description

API responses use a wildcard Access-Control-Allow-Origin: * header, permitting any origin to make credentialed cross-origin requests. Several recommended security headers are absent (X-Content-Type-Options, Referrer-Policy, Permissions-Policy). Server version details are disclosed via the Server: nginx/1.18.0 (Ubuntu) header.

Evidence / Proof of Concept

CORS preflight to /api/v1/me confirmed Access-Control-Allow-Origin: *. Header sampling showed absence of X-Content-Type-Options and Referrer-Policy. Server header disclosed nginx version on all responses.

Steps to Reproduce (Sanitized)

• curl -s -X OPTIONS BASE/api/v1/me -H 'Origin: https://evil.com' — observe Access-Control-Allow-Origin: *
• Inspect response headers — Server: nginx/1.18.0 (Ubuntu) disclosed.

Remediation Guidance

• Restrict CORS to explicitly trusted origins; remove wildcard origins from authenticated API endpoints.
• Add missing security headers: X-Content-Type-Options: nosniff, Referrer-Policy: strict-origin-when-cross-origin,Permissions-Policy.
• Strip or generalise Server and X-Powered-By headers at the reverse proxy layer.

F-API-11 – S3 Bucket CORS Misconfiguration — Low

CVSS Score	3.0 (AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N)
OWASP Mapping	OWASP API8:2023 – Security Misconfiguration | CWE-732
Affected	example-staging.s3.ap-southeast-2.amazonaws.com
Retest Status	Open – Pending Fix

Description

The staging S3 bucket correctly blocks public listing and unauthenticated object access (403 AccessDenied). However, CORS preflight responses allow wildcard origin (*) with PUT and POST methods permitted. If bucket permissions are ever relaxed or pre-signed URLs are used in a browser context, this CORS policy can become exploitable.

Evidence / Proof of Concept

CORS preflight to the bucket returned Access-Control-Allow-Origin: * and Access-Control-Allow-Methods: GET, POST, PUT. Direct PUT and listing attempts returned 403 AccessDenied.

Steps to Reproduce (Sanitized)

- curl -s -X OPTIONS https://locumate-staging.s3.ap-southeast-2.amazonaws.com/ -H 'Origin: https://evil.com' — observe wildcard CORS.
- curl -s -X PUT — returns 403 AccessDenied (currently safe but configuration is permissive).

Remediation Guidance

- Restrict bucket CORS AllowedOrigins to staging application domains only.
- Remove unused AllowedMethods (PUT/POST) if not required for browser-based uploads.
- Ensure S3 Block Public Access is enabled at both bucket and account level.
- If using pre-signed URLs for browser uploads, scope them tightly: short expiry, specific key prefix, content-typeconstraints.

6. CVSS & OWASP Mapping Table

ID	Title	Severity	CVSS v3.1	CVSS Vector (abbreviated)	OWASP Ref
F-WA-01	SQL Injection	High	8.8	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	A1:2017 / CWE-89
F-WA-02	Cross-Site Scripting	High	8.2	AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:L/A:N	A7:2017 / CWE-79
F-WA-03	Session Fixation	High	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	A2:2017 / CWE-384
F-WA-04	Sensitive Data Exposure	High	7.4	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N	A3:2017 / CWE-319
F-WA-05	CSRF	High	8.0	AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N	A8:2013 / CWE-352
F-WA-06	Security Misconfiguration	High	7.2	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	A6:2017 / CWE-16
F-WA-07	Malicious File Upload	High	9.0	AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H	A1:2017 / CWE-434
F-WA-08	Known Vulnerable Components	Medium	6.3	AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N	A9:2017 / CWE-1104
F-WA-09	Banner Disclosure	Low	3.1	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	A6:2017 / CWE-200
F-API-01	BOLA – Job Posts	Critical	9.1	AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N	API1:2023 / CWE-639
F-API-02	BOLA – Chat Access	High	8.1	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	API1:2023 / CWE-639
F-API-03	Broken Func-Level Authz	Critical	9.3	AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:N	API5:2023 / CWE-285
F-API-04	Token Non-Invalidation	High	7.6	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	API2:2023 / CWE-613
F-API-05	Improper Rate Limiting	Medium	5.3	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	API4:2023 / CWE-307
F-API-06	User Enumeration	Medium	5.0	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	API3:2023 / CWE-203
F-API-07	Verbose SQL Errors	High	7.5	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	API8:2023 / CWE-209
F-API-08	Mass Assignment	High	8.0	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	API3:2023 / CWE-915
F-API-09	Pagination Abuse	Medium	4.3	AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N	API4:2023 / CWE-400
F-API-10	Missing Security Headers	Low	3.5	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	API8:2023 / CWE-16
F-API-11	S3 CORS Misconfiguration	Low	3.0	AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	API8:2023 / CWE-732
F-API-12	Undocumented Endpoints	Info	0.0	N/A	API9:2023

7. Remediation Roadmap

IMMEDIATE (0–7 days)

• F-WA-07 – Remove active PHP backdoor from /council/0/15514227260.php.

- F-API-01 – Enforce server-side tenant ownership checks on all job-board endpoints.
- F-API-03 – Apply RBAC middleware to all group-manager routes with deny-by-default.
- F-WA-01 – Parameterise all SQL queries; suppress database error output to clients.

SHORT TERM (1–4 weeks)

- F-WA-02 – Implement context-sensitive output encoding and Content Security Policy.
- F-API-04 – Implement token revocation on logout; switch to short-lived access tokens with refresh token rotation.
- F-API-07 – Disable verbose SQL/exception output; implement centralised error handling.
- F-WA-05 – Add CSRF tokens to all state-changing forms.
- F-API-08 – Implement DTOs/allowlists for job-post creation and update endpoints.
- F-WA-04 – Enforce TLS across all application routes; add HSTS.

MEDIUM TERM (1–3 months)

- F-WA-03 – Implement session ID regeneration on login/logout; add old-password verification.
- F-API-02 – Add participant membership checks to all chat endpoints.
- F-API-05 – Implement rate limiting with lockout on authentication and enumeration endpoints.
- F-API-06 – Neutralise /user-exists response; add CAPTCHA.
- F-WA-06 – Remove debug pages, disable error display, patch PHP/Apache.
- F-API-09 – Cap pagination parameters; enforce per-token rate limits.

ONGOING

- F-WA-08 – Integrate SCA tooling in CI pipeline; upgrade jQuery, Bootstrap, CKEditor.
- F-WA-09 / F-API-10 – Suppress server version headers; add missing security headers.
- F-API-11 – Restrict S3 CORS to trusted origins; remove unused HTTP methods.
- F-API-12 – Audit undocumented endpoints; enforce authentication and document API inventory.
- Establish a continuous vulnerability management programme with quarterly pentests.
- Add automated authorization test suites covering all role × endpoint combinations.

8. Retest Status

The following table tracks the remediation and retest status of all findings. A formal retest should be conducted after critical and high findings are remediated. Retest scope should cover all affected endpoints listed in each finding.

ID	Title	Severity	Status	Target Date	Retested	Result
F-WA-01	SQL Injection	High	Open	Week 1	—	—
F-WA-02	XSS (Reflected & Stored)	High	Open	Week 2	—	—
F-WA-03	Session Fixation	High	Open	Month 1	—	—
F-WA-04	Sensitive Data Exposure	High	Open	Week 2	—	—
F-WA-05	CSRF	High	Open	Week 2	—	—
F-WA-06	Security Misconfiguration	High	Open	Month 1	—	—
F-WA-07	Malicious File Upload / RCE	High	URGENT – Open	Immediate	—	—
F-WA-08	Known Vulnerable Components	Medium	Open	Month 2	—	—
F-WA-09	Banner Disclosure	Low	Open	Month 2	—	—
F-API-01	BOLA – Job Posts	Critical	Open	Immediate	—	—
F-API-02	BOLA – Chat Access	High	Open	Month 1	—	—
F-API-03	Broken Func-Level Authz	Critical	Open	Immediate	—	—

F-API-04	Token Non-Invalidation	High	Open	Week 2	—	—
F-API-05	Improper Rate Limiting	Medium	Open	Month 1	—	—
F-API-06	User Enumeration	Medium	Open	Month 2	—	—
F-API-07	Verbose SQL Errors	High	Open	Week 2	—	—
F-API-08	Mass Assignment	High	Open	Week 2	—	—
F-API-09	Pagination Abuse	Medium	Open	Month 2	—	—
F-API-10	Missing Security Headers	Low	Open	Month 3	—	—
F-API-11	S3 CORS	Low	Open	Month 3	—	—
F-API-12	Undocumented Endpoints	Info	Open	Ongoing	—	—

Retest Guidance

After critical and high remediation is complete, request a focused retest from Pentest Testing Corp. The retest scope should confirm: (1) Cross-tenant access is blocked for all job-board and chat endpoints; (2) Non-authorised roles receive 403/401 consistently on group-manager routes; (3) Bearer tokens are revoked on logout and return 401 on replay; (4) Rate limiting returns 429 with Retry-After on authentication endpoints; (5) File upload accepts only whitelisted extensions and active backdoor is removed; (6) SQL injection payloads return generic errors with no database detail.

9. Appendices

9.1 OWASP API Security Top 10 (2023) Reference

API1:2023	Broken Object Level Authorization (BOLA/IDOR)
API2:2023	Broken Authentication
API3:2023	Broken Object Property Level Authorization
API4:2023	Unrestricted Resource Consumption
API5:2023	Broken Function Level Authorization
API6:2023	Unrestricted Access to Sensitive Business Flows
API7:2023	Server Side Request Forgery
API8:2023	Security Misconfiguration
API9:2023	Improper Inventory Management
API10:2023	Unsafe Consumption of APIs

9.2 OWASP Web Application Top 10 (2017) Reference

A1:2017	Injection
A2:2017	Broken Authentication
A3:2017	Sensitive Data Exposure
A4:2017	XML External Entities (XXE)
A5:2017	Broken Access Control
A6:2017	Security Misconfiguration
A7:2017	Cross-Site Scripting (XSS)

A8:2017	Insecure Deserialization
A9:2017	Using Components with Known Vulnerabilities
A10:2017	Insufficient Logging & Monitoring

9.3 SOC 2 Trust Service Criteria Referenced

CC6.1	Logical and Physical Access Controls — restrict logical access to protected information assets.
CC6.2	Authentication and Credentials — manage credentials to prevent unauthorized access.
CC6.3	Role-Based Access — restrict access based on job function and least privilege.
CC6.6	Security in System Operations — protect against malicious code and unauthorized changes.
CC6.7	Transmission and Disclosure — protect information during transmission.
CC6.8	Prevention and Detection of Unauthorized Software — validate software integrity.
CC7.1	Vulnerability Management — identify, evaluate and remediate vulnerabilities.

CC7.2	Monitoring Controls — monitor system components for anomalies.
CC9.2	Risk Mitigation — assess and mitigate risks from third-party components.

9.4 Key Tested Endpoints (Web App — Selected)

• https://example.com/ · /check_login.php · /get_designations.php
• https://example.com/manage_comp.php · /ajax_getdists2.php · /ajax_getulbs2.php
• https://example.com/cdma_ulbwise_report.php · /update_emp.php · /add_public_rep.php
• https://example.com/sites/admin/videos · /sites/admin/UpdateprofileController/update_userprofile
• https://example.com/council/0/15514227260.php (ACTIVE BACKDOOR — remove immediately)

9.5 Key Tested Endpoints (SaaS / API — Selected)

• POST https://ptestback.example.com.au/api/v1/login
• GET https://ptestback.example.com.au/api/v1/user-exists?login=
• GET/PUT/POST https://ptestback.example.com.au/api/v1/job-board/job-post/{id}
• GET https://ptestback.example.com.au/api/v1/chat/{chat_id}
• GET https://ptestback.example.com.au/api/v1/group-manager/invoices
• GET https://ptestback.example.com.au/api/v1/group-manager/services
• https://locumate-staging.s3.ap-southeast-2.amazonaws.com (S3 bucket)

9.6 About Pentest Testing Corp

Pentest Testing Corp is a specialist offensive security firm providing web application, API, network, and cloud penetration testing services. This report was prepared to support customer security reviews and SOC 2 audit evidence packages.

Website: https://www.pentesttesting.com/