Web Application & API Penetration Test Report

DockLine freight platform
Brightwater Freight, Inc.

Engagement type	Grey-box web application and REST API assessment
Testing window	Ten business days
Report version	1.0 (sample)
Prepared by	Landon Mayo, Principal Application Security Consultant Karman Line AI, LLC · landon.mayo@owasp.org

Sample report: fictional client

This is a writing sample. Brightwater Freight, DockLine, every person, domain, IP address, identifier, credential and finding in this document are invented. Domains use the reserved `.example` top-level domain and public IP addresses come from the documentation ranges. The report is not derived from, and does not describe, any client engagement. It shows the methodology, finding depth and reporting format the author uses on real assessments.

Contents

1 Executive Summary **2**

2 Attack Narrative **2**

3 Scope and Approach **3**

4 Findings Summary **4**

5 Detailed Findings **4**

F-01 JWT algorithm confusion allows forged administrator tokens 4
F-02 Unrestricted file upload leads to remote code execution 5
F-03 Server-side request forgery exposes cloud credentials 6
F-04 Cross-tenant object access in the shipments API 7
F-05 CORS policy reflects any origin with credentials 7
F-06 Stored XSS in shipment notes on the support dashboard 8
F-07 Race condition allows repeated promotion redemption 8
F-08 No throttling on login attempts 9
F-09 Stack traces and version details in error responses 9

6 Appendix: Severity Ratings **9**

1 Executive Summary

Brightwater Freight engaged Karman Line AI to assess the security of DockLine, its multi-tenant freight management platform, from the perspective of an external attacker and of an authenticated customer. Testing covered the customer web application and the public REST API over ten business days, using two tenant accounts and one tenant-administrator account supplied by Brightwater.

The assessment identified **nine findings**: one critical, three high, three medium, one low and one informational. Every finding was confirmed by manual exploitation; none is reported from scanner output alone.

	Critical	High	Medium	Low	Info
Findings	1	3	3	1	1

Overall risk: Critical. An unauthenticated attacker can forge a platform-administrator token (F-01), upload and execute code on the application server (F-02), and from any customer account retrieve the cloud credentials that read every tenant's stored shipping documents (F-03). Separately, any customer can read and modify other customers' shipments (F-04). Together these expose every DockLine tenant's shipment records, bills of lading and customs documents.

What is working well. Database access is parameterized throughout; manual testing and automated injection tooling found no SQL injection. The administrator console enforces FIDO2 multi-factor authentication, TLS configuration is strong, and session cookies are `Secure` and `HttpOnly`. These controls limited the impact of several findings below.

Remediation roadmap

Priority	Action	Findings
Immediate	Pin the JWT signing algorithm; move uploads out of the webroot and block executable types; require IMDSv2 and block link-local egress from the webhook service	F-01, F-02, F-03
30 days	Enforce tenant ownership on every object lookup; replace origin reflection with an allowlist	F-04, F-05
60 days	Encode shipment notes on output and add a content security policy; make promotion redemption atomic	F-06, F-07
90 days	Add login throttling and suppress stack traces in production	F-08, F-09

A remediation retest of all critical and high findings is included within 60 days of this report.

2 Attack Narrative

The findings below are rated individually, but the realistic risk comes from how they chain. Three paths illustrate it.

Path A: no account to full platform compromise

1. The API publishes its RSA public key at `/.well-known/jwks.json`, as is normal. The token verifier, however, trusts the `alg` header supplied by the client. Re-signing a token with HS256, using the public key as the HMAC secret, produced a valid `platform_admin` token without any credentials (F-01).
2. As platform administrator, the tester listed all 214 tenants and opened any shipment. The document-upload endpoint accepted a JSP file labelled `application/pdf` and stored it under a directory that Tomcat serves and executes (F-02). Requesting it ran a harmless proof-of-concept command as the `dockline` service account on `app-01`.
3. Testing stopped at proof of code execution, per the rules of engagement. From that position an attacker would hold the application's database credentials and cloud role.

Path B: one customer account to every tenant's documents

1. From an ordinary customer account, the "test webhook" feature fetched an internal URL and returned the response body. Pointing it at the cloud instance-metadata service returned temporary credentials for the application's IAM role (F-03).
2. Those credentials could list the shared document bucket, which holds every tenant's uploaded bills of lading and customs forms. The tester listed object names to confirm scope and did not download tenant content.
3. Independently, sequential shipment IDs and a missing ownership check let the same customer read and edit other tenants' shipments directly through the API (F-04).

Path C: phishing link to account takeover

The API reflects any `Origin` with credentials allowed (F-05). A customer who visits an attacker-controlled page while signed in to DockLine has their API keys read cross-origin. The attacker then calls the API as that customer without needing their password or MFA.

3 Scope and Approach

In scope	`https://app.dockline.example` (customer web application) `https://api.dockline.example/api/v1` (public REST API, 148 documented operations)
Accounts	Two tenant users in separate tenants; one tenant administrator
Out of scope	Mobile applications; denial-of-service; social engineering; the third-party payment provider; infrastructure outside the application tier
Environment	Production, with written approval and agreed rules of engagement
Rules	No destructive testing; data access limited to the minimum needed to prove impact; critical findings reported to the client within 24 hours of confirmation

Methodology. Testing followed the OWASP Web Security Testing Guide and was scoped to OWASP ASVS Level 2, with PTES phases for planning and reporting. Each API operation was exercised with

every supplied role and with no session, to map authorization boundaries before deeper testing. Automated scanning was used for coverage, and every candidate issue was validated by hand before being reported. Severity uses the CVSS 3.1 base score; the business-impact narrative above explains how findings combine.

Tools. Burp Suite Professional, OWASP ZAP, ffuf, Nuclei, sqlmap, jwt_tool and custom Python scripts.

4 Findings Summary

ID	Finding	Severity	CVSS
F-01	JWT algorithm confusion allows forged administrator tokens	Critical	9.1
F-02	Unrestricted file upload leads to remote code execution	High	8.8
F-03	Server-side request forgery exposes cloud credentials	High	8.5
F-04	Cross-tenant object access in the shipments API	High	8.1
F-05	CORS policy reflects any origin with credentials	Medium	6.5
F-06	Stored XSS in shipment notes on the support dashboard	Medium	5.4
F-07	Race condition allows repeated promotion redemption	Medium	5.3
F-08	No throttling on login attempts	Low	3.7
F-09	Stack traces and version details in error responses	Info	–

5 Detailed Findings

F-01 JWT algorithm confusion allows forged administrator tokens

Severity Critical CVSS 3.1 base score **9.1**
Vector `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N`
Weakness CWE-347: Improper Verification of Cryptographic Signature
Affected All authenticated API operations on `api.dockline.example`

Description. DockLine issues RS256-signed JSON Web Tokens and publishes the matching public key at `/.well-known/jwks.json`. The verification code passes the token's own `alg` header to the JWT library together with the key. When a token declares `HS256`, the library treats the public key's PEM text as an HMAC secret. Because that key is public, anyone can sign tokens the API accepts, with any subject, tenant and role.

Evidence. A token re-signed with HS256 and `"role":"platform_admin"` was accepted:

```
GET /api/v1/admin/tenants HTTP/1.1
Host: api.dockline.example
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
     eyJzdWIiOiJ1c3JfdGVzdCIsInJvbGUiOiJwbGF0Zm9ybV9hZG1pbiJ9.[signature]

HTTP/1.1 200 OK
Content-Type: application/json
```

```
{"total":214,"tenants":[{"id":"tnt_0001","name":"[redacted]"}, ...]}
```

Impact. Complete loss of authentication and authorization for the API. An attacker needs no account and can act as any user, including platform administrators.

Remediation.

- Fix the accepted algorithm server-side: verify with `algorithms=["RS256"]` and reject any token whose header differs.
- Load the verification key as an RSA key object, not as a string, so it cannot be reused as an HMAC secret.
- Rotate the signing key pair after the fix and invalidate existing sessions.
- Add a regression test that submits HS256 and `none` tokens and expects 401.

F-02 Unrestricted file upload leads to remote code execution

Severity	**High** CVSS 3.1 base score **8.8**
Vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`
Weakness	CWE-434: Unrestricted Upload of File with Dangerous Type
Affected	`POST /api/v1/shipments/{id}/documents`; files served from `https://app.dockline.example/uploads/`

Description. The document endpoint checks only the client-supplied `Content-Type` header. It keeps the original filename and writes the file into a directory inside the Tomcat webroot, where `.jsp` files are compiled and executed on request.

Evidence. A JSP file declared as a PDF was stored and executed. The proof-of-concept only printed the service identity and hostname:

```
POST /api/v1/shipments/SHP-200117/documents HTTP/1.1
Host: api.dockline.example
Content-Type: multipart/form-data; boundary=----kl

------kl
Content-Disposition: form-data; name="file"; filename="bol_200117.jsp"
Content-Type: application/pdf

<% /* proof of concept: prints identity only */ %> ...
------kl--

HTTP/1.1 201 Created
{"url":"https://app.dockline.example/uploads/bol_200117.jsp"}

GET /uploads/bol_200117.jsp HTTP/1.1
Host: app.dockline.example

HTTP/1.1 200 OK
uid=1001(dockline) gid=1001(dockline) groups=1001(dockline)
app-01
```

Impact. Any authenticated user (or, through F-01, any attacker) can run commands on the application server with the service account's access to the database and cloud role.

Remediation.

- Store uploads in object storage or a directory outside the webroot, under server-generated random names.
- Allowlist the expected extensions (`pdf`, `png`, `jpg`) and verify file signatures server-side.
- Serve documents through a download endpoint with `Content-Disposition: attachment` and `X-Content-Type-Options: nosniff`.
- Disable JSP execution for any path that holds user content, and scan uploads before they are available.

F-03 Server-side request forgery exposes cloud credentials

Severity High CVSS 3.1 base score **8.5**
Vector `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:L/A:N`
Weakness CWE-918: Server-Side Request Forgery
Affected `POST /api/v1/webhooks/test`

Description. The "test webhook" feature requests a customer-supplied URL from the server and returns the response body. It does not restrict destinations, so internal and link-local addresses are reachable. The instance-metadata service accepts unauthenticated (IMDSv1) requests.

Evidence. The webhook test returned the role's temporary credentials, which listed the shared bucket:

```
POST /api/v1/webhooks/test HTTP/1.1
Host: api.dockline.example
Authorization: Bearer [tenant user token]
Content-Type: application/json

{"url":"http://169.254.169.254/latest/meta-data/iam/security-credentials/dockline-app-role"}

HTTP/1.1 200 OK
{"status":200,"body":"{\"Code\":\"Success\",\"AccessKeyId\":\"ASIA[redacted]\",\"SecretAccessKey\":\"[
    redacted]\",\"Token\":\"[redacted]\",\"Expiration\":\"[redacted]\"}"}

$ aws s3 ls s3://dockline-prod-documents/ --profile ssrf-test
                           PRE tnt_0001/
                           PRE tnt_0002/
                           ...  (214 tenant prefixes; object contents not downloaded)
```

Impact. A single customer account obtains the application's cloud identity and can read every tenant's stored documents. The scope change in the vector reflects access beyond the vulnerable component.

Remediation.

- Require IMDSv2 (`HttpTokens=required`) on all application instances.
- Send webhook traffic through an egress proxy that denies link-local, loopback and private ranges, validating the resolved address rather than the hostname.

- Return only the status code from test deliveries, never the response body.
- Reduce the application role to the specific prefixes and actions it needs.

F-04 Cross-tenant object access in the shipments API

Severity High CVSS 3.1 base score **8.1**
Vector `CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N`
Weakness CWE-639: Authorization Bypass Through User-Controlled Key
Affected `GET` and `PATCH /api/v1/shipments/{id}`

Description. Shipment IDs are sequential, and the shipment endpoints check that the caller is authenticated but not that the shipment belongs to the caller's tenant.

Evidence. The tester's tenant-B account read and then changed a tenant-A shipment (a test record created by the client for this purpose):

```
PATCH /api/v1/shipments/SHP-104422 HTTP/1.1
Host: api.dockline.example
Authorization: Bearer [tenant B token]
Content-Type: application/json

{"consignee":{"address_line1":"KLAI TEST - AUTHZ CHECK"}}

HTTP/1.1 200 OK
{"id":"SHP-104422","tenant_id":"tnt_0042","consignee":{"address_line1":"KLAI TEST - AUTHZ CHECK", ...}}
```

Impact. Any customer can read other customers' shipments and redirect deliveries by changing consignee details.

Remediation. Enforce tenant ownership in a shared data-access layer so every object lookup is filtered by the caller's tenant; return 404 for objects in other tenants; add automated authorization tests that replay each object endpoint with a second tenant's token. Non-sequential identifiers are useful defense in depth but are not a fix on their own.

F-05 CORS policy reflects any origin with credentials

Severity Medium CVSS 3.1 base score **6.5**
Vector `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N`
Weakness CWE-942: Permissive Cross-domain Policy with Untrusted Domains
Affected All endpoints on `api.dockline.example`

Description. The API copies the request's `Origin` header into `Access-Control-Allow-Origin` and sets `Access-Control-Allow-Credentials: true`. The session cookie is `SameSite=None`, so it is sent on cross-site requests.

Evidence. A credentialed request from a foreign origin was answered with the user's API key:

```
GET /api/v1/users/me/api-keys HTTP/1.1
Host: api.dockline.example
Origin: https://attacker.example
Cookie: dl_session=[victim session]

HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://attacker.example
Access-Control-Allow-Credentials: true
{"keys":[{"id":"key_31","secret":"dlk_live_[redacted]"}]}
```

Impact. A signed-in user who opens an attacker's page loses their API keys, which grant API access without a password or MFA (attack path C).

Remediation. Allow only DockLine's own origins; set the session cookie to `SameSite=Lax`; show API key secrets once at creation and never return them again.

F-06 Stored XSS in shipment notes on the support dashboard

Severity	Medium CVSS 3.1 base score **5.4**
Vector	`CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N`
Weakness	CWE-79: Improper Neutralization of Input During Web Page Generation
Affected	Shipment `notes` field, rendered in the internal support dashboard

Description. Customer-entered shipment notes are rendered as raw HTML in the support dashboard. A note containing `<img src=x onerror=...>` ran script in a support agent's browser when the shipment was opened.

Impact. Script runs with the support agent's privileges in that page. Session theft is blocked by `HttpOnly` cookies, but the script can perform any action the agent can, such as editing account details.

Remediation. Render notes as text with contextual output encoding; if rich text is required, sanitize with an allowlist library; deploy a content security policy that blocks inline script.

F-07 Race condition allows repeated promotion redemption

Severity	Medium CVSS 3.1 base score **5.3**
Vector	`CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:H/A:N`
Weakness	CWE-362: Concurrent Execution Using Shared Resource with Improper Synchronization
Affected	`POST /api/v1/billing/promotions/redeem`

Description. Single-use promotion codes are checked and then applied in separate steps without a lock. Twenty requests sent in parallel with the same code applied the $50 credit six times.

Impact. Customers can multiply account credits. The issue was confirmed with a client-issued test code and the resulting credits were reversed by Brightwater.

Remediation. Make redemption atomic: a unique constraint on (account, promotion) plus a transaction that inserts the redemption before applying credit; accept an idempotency key on billing endpoints.

F-08 No throttling on login attempts

Severity Low CVSS 3.1 base score **3.7**
Vector `CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N`
Weakness CWE-307: Improper Restriction of Excessive Authentication Attempts
Affected `POST /api/v1/auth/login`

Description. One thousand failed logins against a test account in four minutes produced no lockout, delay or challenge. MFA is optional for customer users.

Remediation. Add per-account and per-source rate limits with progressive delay, alert on spraying patterns, and encourage or require MFA for customer administrators.

F-09 Stack traces and version details in error responses

Severity Info CVSS 3.1 base score –
Vector `Not scored`
Weakness CWE-209: Generation of Error Message Containing Sensitive Information
Affected Error responses from both hosts

Description. Malformed requests return full Java stack traces that reveal framework and library versions and internal package names. This speeds an attacker's research but is not exploitable on its own.

Remediation. Return a generic error with a correlation ID and log detail server-side only.

6 Appendix: Severity Ratings

Severity follows the CVSS 3.1 base score. The attack narrative explains where combined findings exceed the individual ratings.

Severity	CVSS	Meaning
Critical	9.0 – 10.0	Directly exploitable with severe impact; fix immediately
High	7.0 – 8.9	Significant data exposure or system compromise; fix urgently
Medium	4.0 – 6.9	Exploitable under conditions or with limited impact; schedule a fix
Low	0.1 – 3.9	Minor impact or hard to exploit; fix in normal maintenance
Info	–	No direct risk; improves defense in depth