SAMPLE DOCUMENT

MM

PENETRATION TEST · WEB APPLICATION AND API

Penetration Test Report

Customer web application and API · Technical security assessment

Example Client, Ltd.

Reference: MM-2026-PT-0142 · Version: 1.0 · Date: 12 May 2026
Scope: app.example-client.com (staging environment, synthetic data)
Standards: OWASP WSTG 4.2 · PTES · Classification: Confidential
Author: Martín Martín · mmartin.me

NOTICE

Confidentiality and nature of this document

Sample document

This report is a **demonstration sample**. It does not correspond to any real client or system. The organisation (Example Client, Ltd.), the domain (`example-client.com`), the data and the screenshots are fictional, and the IP addresses belong to the ranges reserved for documentation by RFC 5737 (`203.0.113.0/24`). Its only purpose is to show the format, level of detail and rigour of a penetration test report.

Real reports delivered to clients are covered by a non-disclosure agreement (NDA) and are not shared. This document takes the place of that sample without exposing any client.

Confidentiality clause

The information in a report of this kind describes concrete, exploitable security weaknesses. Distribution is limited to the people in the organisation with direct responsibility for remediating the findings and for the relationship with the certification auditor. Reproduction in whole or in part beyond that circle requires express authorisation.

Contents

1 Introduction

1.1 Context and objectives

Example Client, Ltd. commissions a technical security assessment of its customer-facing web application and the API behind it, as part of its preparation for ISO/IEC 27001 certification and its SOC 2 programme.

The objective is twofold. First, to identify and demonstrate real vulnerabilities that an attacker could exploit against the application and its users. Second, to provide technical evidence that the auditor can accept as proof that the organisation subjects its systems to periodic security testing, with findings, prioritisation and verification of remediation.

An automated scanner does not cover the second objective. It flags known weak configurations, but it does not reason about business logic or about the access boundaries between clients. The main finding of this assessment (HALL-01) is exactly of that kind: no automated tool would have found it.

1.2 Agreed scope

Primary asset	`https://app.example-client.com (web application + REST API /api/v1)`
Environment	Staging, a functional replica of production, with synthetic data. No testing was run against the production environment.
Test type	Grey box. Two customer accounts from different organisations and one standard user account were provided, with no administration credentials.
Perspectives	Unauthenticated and authenticated (standard customer).
Testing window	5 to 9 May 2026, within the agreed hours.
Out of scope	Underlying network infrastructure, denial-of-service attacks, social engineering and phishing, third-party systems (payment gateway, email).

Any testing against the production environment is carried out only under a specific contract that defines the window, safeguards, data involved and responsibilities. By default the work is done on staging with synthetic data.

1.3 Methodology and standards

Testing follows the **OWASP Web Security Testing Guide (WSTG 4.2)** and the **PTES** framework (Penetration Testing Execution Standard), combining manual review with tool support. The work is organised into reconnaissance, attack-surface mapping, authentication and authorisation testing, business-logic testing, and validation and chaining of findings.

Each finding is related to the controls of **ISO/IEC 27001:2022 Annex A** and to the **SOC 2 Trust Services Criteria** (Common Criteria) it helps to evidence. That mapping appears in the table in section 3 and in each finding. It is indicative, not an exhaustive statement about every control in the management system, and is meant to point out the controls this test helps to evidence.

1.4 Severity scale

The severity of each finding is expressed as a qualitative category (Critical, High, Medium, Low, Informational), backed by a **CVSS 3.1** score with its full vector. The category drives the business decision; the vector gives the technical traceability the auditor can review.

Severity	CVSS 3.1	Interpretation
CRITICAL	9.0 – 10.0	Direct compromise of data or the system. Fix immediately.
HIGH	7.0 – 8.9	Serious impact and viable exploitation. Fix as a priority.
MEDIUM	4.0 – 6.9	Appreciable impact or conditional exploitation. Plan the fix.
LOW	0.1 – 3.9	Limited impact. Defence-in-depth hardening.
INFORMATIONAL	0.0	Observation with no direct security impact.

2 Executive summary

The application has a serious tenant-isolation problem that lets an authenticated user access the billing data of other organisations. It is the finding that defines this assessment and it requires immediate remediation.

The assessment identified **six findings**: one critical, two high, two medium and one low. Most of them concentrate in access control, that is, in who can do what inside the application. This pattern is common in applications that have grown fast: features are built assuming the previous layer already checked authorisation, and no one checks it.

1	2	2	1	6
CRITICAL	HIGH	MEDIUM	LOW	TOTAL

Business risk

The critical finding (HALL-01) exposes personal and financial data of the platform's clients to any registered user. In a multi-tenant application this is the worst possible confidentiality outcome: it breaks the boundary that separates one client from another. Beyond the reputational damage, it is a personal-data breach with regulatory implications (GDPR).

The two high findings allow, respectively, a normal user to become an administrator (HALL-02) and a customer to run code in the browser of the administration staff (HALL-03). Either one turns a low-privilege account into effective control of the platform.

Remediation priority

- **Immediate:** HALL-01 (tenant isolation) and HALL-02 (privilege escalation).
- **Short term:** HALL-03 (stored XSS) and HALL-04 (credit through negative quantity).
- **Planned:** HALL-05 (enumeration and rate limiting) and HALL-06 (headers and cookies).

None of the findings requires redesigning the application. All are fixed by applying server-side validation and authorisation checks, as detailed in each section. A verification (retest) is included to confirm the fixes before the audit.

3 Findings summary

The table below summarises the findings, their severity and the controls each one helps to evidence. The "Status" column reflects the situation at the time this report is issued.

ID	Title	Severity	CVSS	Status	ISO 27001:2022	SOC 2 (CC)
HALL-01	Access to other clients' invoices (IDOR)	CRITICAL	9.1	Open	A.5.15 · A.8.3	CC6.1 · CC6.3
HALL-02	Privilege escalation to administrator	HIGH	8.1	Open	A.5.15 · A.8.2	CC6.1 · CC6.2
HALL-03	Stored XSS in the administration console	HIGH	7.6	Open	A.8.28 · A.8.26	CC6.8 · CC7.2
HALL-04	Arbitrary credit through negative quantity	MEDIUM	6.5	Open	A.8.26 · A.8.29	CC7.1 · CC8.1
HALL-05	Account enumeration and missing rate limiting	MEDIUM	5.3	Open	A.8.5	CC6.1 · CC6.6
HALL-06	Security headers and cookie attributes	LOW	3.1	Open	A.8.9	CC6.6

Control reference: A.5.15 Access control · A.8.2 Privileged access rights · A.8.3 Information access restriction · A.8.5 Secure authentication · A.8.9 Configuration management · A.8.26 Application security requirements · A.8.28 Secure coding · A.8.29 Security testing in development and acceptance. SOC 2 mapping is to the Common Criteria (Security) and is indicative.

4 Detailed findings

HALL-01

Access to other clients' invoices (IDOR)

CRITICAL

CVSS 3.1: 9.1
AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N

SUMMARY

The endpoint `GET /api/v1/invoices/{id}` returns an invoice by its numeric identifier without checking that the invoice belongs to the authenticated user's organisation. Any registered customer (sign-up is open) can walk the identifiers and read the invoices of the other organisations on the platform.

IMPACT

Break of tenant isolation in a multi-tenant application. Any user reads third-party billing data: legal name, tax ID, amounts, line items and billing address. It is a personal-data breach (GDPR) and the worst possible confidentiality outcome for this kind of platform. Sequential enumeration of identifiers allows the full billing database to be extracted systematically.

PROOF OF CONCEPT

Authenticated as a user of organisation A (test account `demo-a`), an own invoice is requested and then an adjacent invoice belonging to organisation B:

```
# Own invoice (organisation A): expected
GET /api/v1/invoices/1042 HTTP/1.1
Host: app.example-client.com
Authorization: Bearer <token-demo-a>

HTTP/1.1 200 OK
{"id":1042,"organization":"A","tax_id":"GB000000001","total":1210.00, ...}

# Invoice of another organisation: should NOT be accessible
GET /api/v1/invoices/1043 HTTP/1.1
Host: app.example-client.com
Authorization: Bearer <token-demo-a>

HTTP/1.1 200 OK
{"id":1043,"organization":"B","tax_id":"GB000000002","total":3480.50,
 "client":"Another Company, Ltd.","address":"1 Example Street, London", ...}
```

[Screenshot: 200 response with organisation B billing data, obtained with organisation A's token]

The same pattern repeats across the identifier range. The response does not vary by owning organisation: the server only checks that the token is valid, not that the resource belongs to the requester.

REMEDIATION

Enforce the ownership check on the server, in the data-access layer, not in the interface. The query must filter by the token's organisation, not only by the identifier:

```
// Instead of: look up by id
SELECT * FROM invoices WHERE id = :id;

// Always filter by the authenticated user's organisation
SELECT * FROM invoices WHERE id = :id AND organization_id = :org_from_token;
```

- Derive the organisation from the session token on the server, never from a client parameter.
- Return 404 (not 403) for a resource of another organisation, so its existence is not confirmed.
- Apply the same ownership filter to every endpoint that takes a resource identifier, not only invoices.
- Add automated cross-organisation authorisation tests to the integration test suite.

HALL-02

Privilege escalation to administrator

HIGH

CVSS 3.1: 8.1
AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N

SUMMARY

The profile update endpoint `PUT /api/v1/profile` accepts a `role` field sent by the client and applies it without validation. A standard user can set themselves the administrator role and gain access to the administration console.

IMPACT

A low-privilege user gains full control of the platform: user management, access to configuration and to the data of every organisation. Combined with open sign-up, anyone on the internet can become an administrator.

PROOF OF CONCEPT

```
# The "role" field travels in the request body and is accepted as-is
PUT /api/v1/profile HTTP/1.1
Host: app.example-client.com
Authorization: Bearer <standard-user-token>
Content-Type: application/json

{"name":"Demo","role":"admin"}

HTTP/1.1 200 OK
{"name":"Demo","role":"admin"}

# From here, the administration console is reachable
GET /admin HTTP/1.1  ->  HTTP/1.1 200 OK
```

[Screenshot: administration console reachable after changing the role from a standard account]

REMEDIATION

- Never accept the role or any authorisation attribute from the client. The role is set and changed only on the server, through a controlled administrative flow.
- Explicitly discard unexpected fields in the request body (allow-list), rather than assigning the whole received object.
- Check the user's effective role on the server at each access to administration features.

HALL-03

Stored XSS in the administration console

HIGH

CVSS 3.1: 7.6
AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:L/A:N

SUMMARY

The subject of support tickets is stored without neutralisation and rendered without escaping in the ticket queue of the administration console. A customer can put code in the subject that runs in the browser of the administration staff when they open the queue.

IMPACT

The code runs in the application's origin and in an administrator's session, a higher privilege than the attacker's. It allows the session to be stolen, actions to be taken as the administrator, or chaining with HALL-02. Because it crosses the privilege boundary (customer to administrator), the scope is "changed" (S:C) in the CVSS vector.

PROOF OF CONCEPT

```
# A customer creates a ticket with code in the subject
POST /api/v1/tickets HTTP/1.1
Host: app.example-client.com
Content-Type: application/json

{"subject":"<img src=x onerror=alert(document.domain)>",
 "message":"Test enquiry"}

# On opening the ticket queue, the admin's browser runs the code
```

[Screenshot: alert(document.domain) executing in the administration console when the tickets are listed]

REMEDIATION

- Escape the output by context at render time (HTML-encode the subject and any user-supplied data). This is the primary fix.
- Validate the input on the server as reinforcement, not as the only defence.
- Apply a Content Security Policy (CSP) that limits inline script execution in the administration console.

HALL-04

Arbitrary credit through negative quantity

MEDIUM

CVSS 3.1: 6.5
AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N

SUMMARY

The checkout process accepts a negative quantity on a cart line. The resulting total is negative and the difference is applied as account credit. The server trusts the total computed on the client instead of recomputing it.

IMPACT

Direct financial fraud. A user can generate arbitrary credit and reduce the amount payable below its real value. It does not compromise confidentiality, but it does compromise the integrity of the platform's financial data.

PROOF OF CONCEPT

```
# A line with a negative quantity drops the total below zero
POST /api/v1/cart/line HTTP/1.1
Host: app.example-client.com
Content-Type: application/json

{"product":123,"quantity":-5}

HTTP/1.1 200 OK
{"total":-245.00,"credit_applied":245.00}
```

[Screenshot: cart summary with a negative total and credit applied to the account]

REMEDIATION

- Validate on the server that the quantity is an integer greater than zero before processing the line.
- Always recompute the total on the server from trusted prices. Do not accept totals or amounts computed by the client.
- Define business invariants (the total is never negative) and log any attempt to break them for monitoring.

HALL-05

MEDIUM

Account enumeration and missing rate limiting

CVSS 3.1: 5.3
AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

SUMMARY

The password-reset endpoint `POST /api/v1/reset-password` responds differently depending on whether the email exists, which allows valid accounts to be enumerated. It also applies no rate limiting, so enumeration and brute-force attempts meet no obstacle.

IMPACT

An attacker can confirm which email addresses have an account on the platform and use that list for targeted attacks. Without rate limiting, enumeration is bulk and, if the reset token were weak, it would ease account takeover.

PROOF OF CONCEPT

```
# Email with an account
POST /api/v1/reset-password  {"email":"exists@example.com"}
HTTP/1.1 200 OK  {"message":"We have sent you an email"}

# Email without an account: the response gives it away
POST /api/v1/reset-password  {"email":"nouser@example.com"}
HTTP/1.1 404 Not Found  {"error":"No account with that email"}

# No rate limit: hundreds of requests without a block
```

REMEDIATION

- Always respond the same way, whether or not the account exists ("if the email has an account, you will get a message").
- Apply rate limiting per IP and per account, and a challenge (CAPTCHA) above a threshold.
- Issue high-entropy, single-use, short-lived reset tokens.

HALL-06

LOW

Security headers and cookie attributes

CVSS 3.1: 3.1
AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N

SUMMARY

The application's responses do not include the `Strict-Transport-Security` header or a Content Security Policy, and the session cookie is issued without the `SameSite` attribute and, on one route, without `Secure`. These are defence-in-depth weaknesses, with no direct exploitation on their own.

IMPACT

They reduce the safety margins against protocol-downgrade and request-forgery attacks in specific scenarios. They do not compromise the application in isolation, but they are worth fixing for hygiene and because they are reviewed in audits.

PROOF OF CONCEPT

```
# Response without HSTS or CSP; cookie without SameSite
HTTP/1.1 200 OK
Set-Cookie: session=...; Path=/; HttpOnly
# missing: Strict-Transport-Security, Content-Security-Policy,
#          and the SameSite attribute on the session cookie
```

REMEDIATION

- Add `Strict-Transport-Security` with a reasonable duration and, once assessed, inclusion in the preload list.
- Define a Content Security Policy (CSP) tuned to the application.
- Issue the session cookie with `SameSite=Lax` (or `Strict`), `Secure` and `HttpOnly` on every route.

5 Testing coverage

This section summarises the test categories run over the scope, following the OWASP Web Security Testing Guide. It records the coverage: what was reviewed, whether or not findings were identified.

Category (OWASP WSTG)	Status	Result
Information gathering	Tested	No findings
Configuration and deployment	Tested	HALL-06
Identity management and registration	Tested	HALL-02
Authentication	Tested	HALL-05
Authorisation and access control	Tested	HALL-01, HALL-02
Session management	Tested	HALL-06 (cookie attributes)
Input and output validation	Tested	HALL-03
Business logic	Tested	HALL-04
Cryptography in transit (TLS)	Tested	No findings
Error handling	Tested	No findings
REST API security	Tested	HALL-01, HALL-02, HALL-05
Client-side	Tested	HALL-03

"No findings" means the category was reviewed and no reportable weaknesses were identified, not that the risk is zero.

6 Positive observations

Not everything reviewed shows weaknesses. This section records the controls that behaved correctly, to give a balanced picture of the security posture and to flag the practices worth keeping.

- **Password storage.** Passwords are stored with a strong, salted hash function. No plaintext passwords or obsolete algorithms were identified.
- **Encryption in transit.** The service enforces HTTPS with TLS 1.2 or higher and a suitable cipher suite. No obsolete protocols or ciphers were detected.
- **No SQL injection.** The queries reviewed use parameterised statements. No SQL injection was identified anywhere in the scope.
- **Session handling on logout.** Logout invalidates the token on the server, not only on the client.
- **CSRF protection.** State-changing forms include a valid anti-CSRF token that is verified on the server.
- **Controlled error messages.** Errors do not reveal internal stack traces, file paths or component versions.

A Verification (retest)

Remediation of the findings is confirmed with a follow-up verification, included in the scope of the work. Its goal is for the organisation to reach the audit with evidence that the identified weaknesses have been closed.

How it works

- The organisation reports which findings it has fixed and provides access to the same staging environment.
- Each finding is retested with the same procedure it was identified with, not only the declared fix.
- Each finding moves to one of three states: **Fixed**, **Partially fixed** (with the detail of what is missing) or **Not fixed**.
- A verification appendix is issued with the updated status table, which accompanies the original report as evidence for the auditor.

Scope and timing

Includes	Retest of all the findings in the report, in the same environment.
Delivery time	3 working days from access to the fixed environment.
Result	Verification appendix with statuses and, where relevant, guidance on incomplete fixes.

End of sample report. Demonstration document by Martín Martín (mmartin.me). Neither the organisation nor the data are real.