Vectis
CONSULTING

PENETRATION TEST · REPORT

Penetration Test Report

Exemple SA

Penetration test of the Exemple web application and API

REFERENCE	VERSION	DATE	CLASSIFICATION
RPT-2026-0001	1.0	17 December 2026	CONFIDENTIAL

Vectis Consulting · contact@vectis-consulting.cloud · vectis-consulting.cloud

Document control

Client	Exemple SA
Engagement	ENG-2026-0001 — Penetration test of the Exemple web application and API
Document	RPT-2026-0001, version 1.0, 17 December 2026
Classification	**Confidential** — distribution limited to the recipients below
Performed by	Vectis Consulting
Review and approval	Management, Vectis Consulting
Authorisation	AUT-2026-0001 — scope version 2 SHA-256 `6b1e245e1b73353f2266632ead26313f5b4f734df5ba97e41d585874aa615399`

Version history

VERSION	DATE	AUTHOR	CHANGES
0.9	14 December 2026	Vectis Consulting	Draft report
1.0	17 December 2026	Vectis Consulting management	Expert review, approval and release

Distribution list

RECIPIENT	ORGANISATION
Jeanne Exemple Head of IT Security	Exemple SA
Management	Vectis Consulting

> **Confidentiality**
>
> This report describes exploitable vulnerabilities in the systems of Exemple SA. It is intended for the recipients on the distribution list only. Share it only with people who need to know, through a protected channel. Vectis Consulting keeps the final report for 5 years after the end of the engagement and deletes the engagement's evidence and working files no later than 90 days after the end of the engagement.

Contents

1 **Executive summary** 4

2 **Scope and methodology** 5

3 **Summary of findings** 7

4 **Detailed findings** 8

VUL-01 SQL injection in the sort parameter of the products API 8

VUL-02 Access to other customers' invoices by changing the identifier 10

VUL-04 No limit on login attempts 11

VUL-03 Reflected cross-site scripting (XSS) in the search page 12

VUL-05 Missing Strict-Transport-Security header 13

VUL-06 Disclosure of server component versions 14

5 **Remediation plan** 15

A **Appendix — Timeline** 16

B **Appendix — Tools** 17

C **Appendix — Glossary** 18

1 Executive summary

From 8 to 10 December 2026, Vectis Consulting performed a grey-box penetration test of Exemple SA's web application and API. The goal was to identify vulnerabilities that an external attacker could exploit, with or without a user account.

The overall risk level is **critical**. An SQL injection reachable without authentication makes the application database readable, and a broken access control lets any authenticated customer read other customers' invoices. Both vulnerabilities expose personal and business data: they must be fixed first, and the logs reviewed to rule out earlier exploitation.

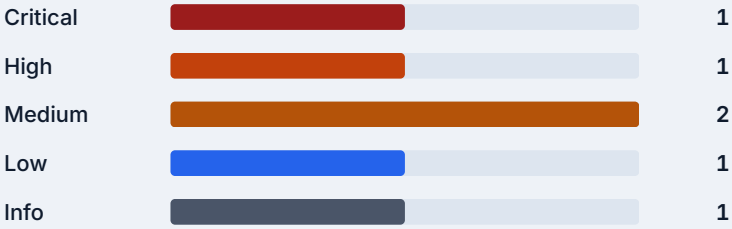
The other findings are less severe but make targeted attacks against users easier. The TLS configuration and the protection of session cookies are sound.

OVERALL RISK LEVEL

Critical

6 vulnerabilities and findings

BREAKDOWN BY SEVERITY



Key vulnerabilities

CRITICAL **VUL-01 — SQL injection in the sort parameter of the products API**
The `sort` parameter is inserted as-is into the SQL query: the database can be read without authentication.

HIGH **VUL-02 — Access to other customers' invoices by changing the identifier**
The API returns any invoice to any authenticated user who knows its sequential identifier.

Priority recommendations

1. Fix VUL-01 and VUL-02 without delay, then check the logs to make sure they have not been exploited.
2. Use parameterised queries everywhere and a central, server-side authorisation check for every object accessed.
3. Limit login attempts and offer multi-factor authentication.
4. Add automated security tests (static analysis, authorisation tests) to the deployment pipeline.
5. Schedule a retest after remediation to confirm that the fixes are effective.

Strengths observed

- Strong TLS configuration: only TLS 1.2 and 1.3 are accepted, with modern cipher suites.
- Session cookies have the HttpOnly, Secure and SameSite=Lax attributes.
- API error messages disclose neither stack traces nor SQL queries.

2 Scope and methodology

2.1 Tested scope

#	TARGET	TYPE	NOTES
1	`https://app.example.com`	URL	Web application (user interface)
2	`api.example.com`	Domain name	REST API v2, HTTPS only
3	`203.0.113.10`	IPv4 address	Front-end server, ports 80 and 443

Exclusions

#	EXCLUSION	NOTES
1	`https://app.example.com/billing/`	Payment module
2	`203.0.113.25`	Mail server

2.2 Testing conditions

Test type	Grey box (with test accounts provided by the Client)
Authorised window	8 December 2026 at 09:00 → 10 December 2026 at 18:00
Actual testing	8 December 2026 at 09:02 → 10 December 2026 at 17:41
Source IP address	`198.51.100.7`
Test accounts	client-a ("customer" role), client-b ("customer" role)
Items created during testing	Saved search "vectis-test" on the client-a account: deleted at the end of testing.
Authorisation	AUT-2026-0001, scope version 2

Times are given in Brussels time.

2.3 Methodology

The tests follow the public OWASP and PTES frameworks:

- **OWASP Web Security Testing Guide (WSTG)** — catalogue of application tests, whose identifiers appear in the references of the findings;
- **OWASP Application Security Verification Standard (ASVS)** — application security requirements;
- **Penetration Testing Execution Standard (PTES)** — how a penetration test is conducted.

The engagement has four phases: reconnaissance and mapping of the authorised targets; automated and manual vulnerability research; non-destructive exploitation, limited to demonstrating the impact; analysis, validation and rating of the results. Vulnerabilities are rated with version 4.0 of the Common Vulnerability Scoring System (CVSS), published by FIRST.

2.4 Testing approach

> Testing combined automated tools, including AI-based tools, with manual verification. The scope was enforced at network level, independently of the tools: only the authorised targets could be reached, during the authorised window, from the declared source IP address and with a capped rate. Actions that could put the tested systems at risk were performed only with the expert's prior approval, and every action was logged.
>
> Every finding in this report was reviewed and validated by an expert before issue.

2.5 Severity scale

Severity derives from the CVSS 4.0 base score, which measures technical exploitability and impact. It does not take into account the business criticality of the data concerned, which the Client is best placed to assess.

SEVERITY	CVSS 4.0 SCORE	MEANING
CRITICAL	9.0 – 10.0	Easy exploitation and major impact: compromise of data or systems. Address immediately.
HIGH	7.0 – 8.9	Significant impact or likely exploitation. Address in the short term.
MEDIUM	4.0 – 6.9	Limited impact or exploitation subject to conditions. Plan the fix.
LOW	0.1 – 3.9	Low impact or difficult exploitation. Address with planned changes.
INFO	—	Finding with no direct security impact, or a good practice not applied.

2.6 Limitations

A penetration test is limited in time, scope and techniques. Its results describe the state of the systems during the test window; they do not guarantee the absence of other vulnerabilities. Limitations specific to this engagement:

- The payment module (https://app.example.com/billing/) and the mail server (203.0.113.25) were excluded from the scope.
- Only the "customer" role was tested, with two test accounts; the administrator role was not covered.
- Denial-of-service testing and social engineering were not part of the engagement.

3 Summary of findings

The table below lists the vulnerabilities and findings, sorted by decreasing severity.

ID	TITLE	SEVERITY	CVSS	AFFECTED ASSET
VUL-01	SQL injection in the sort parameter of the products API	CRITICAL	9.3	https://api.example.com/v2/products
VUL-02	Access to other customers' invoices by changing the identifier	HIGH	7.1	https://api.example.com/v2/invoices/{id}
VUL-04	No limit on login attempts	MEDIUM	6.3	https://api.example.com/v2/session
VUL-03	Reflected cross-site scripting (XSS) in the search page	MEDIUM	5.1	https://app.example.com/search?q=
VUL-05	Missing Strict-Transport-Security header	LOW	2.3	https://app.example.com
VUL-06	Disclosure of server component versions	INFO	—	https://app.example.com, api.example.com

4 Detailed findings

Every vulnerability was reproduced and validated before being reported. Any personal data present in the evidence has been masked.

VUL-01

SQL injection in the sort parameter of the products API

Severity	CRITICAL **9.3** / 10 (CVSS 4.0)
CVSS 4.0 vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N`
Affected asset	`https://api.example.com/v2/products`
References	CWE-89 · OWASP WSTG-INPV-05

DESCRIPTION

The `sort` parameter of the `GET /v2/products` endpoint is inserted without validation into the `ORDER BY` clause of the SQL query. By placing a conditional expression in it, an unauthenticated attacker changes the order of the results depending on a value of their choice, which makes it possible to extract the database content character by character (boolean inference).

In line with the rules of engagement, exploitation was limited to reading the database user name (`shop_app`) and the server version (PostgreSQL 15). No data was extracted or modified.

IMPACT

An attacker can read all data available to the application account: customer accounts, addresses, orders and password hashes. Depending on that account's privileges, they may also modify or delete data. The vulnerability can be exploited from the Internet without an account, and public tools automate its exploitation.

EVIDENCE

Evidence 1. True condition: products are sorted by name

```
GET /v2/products?sort=(CASE+WHEN+(SELECT+substr(current_user,1,1))='s'+THEN+name+ELSE+price+END) HTTP/1.1
Host: api.example.com
Accept: application/json

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

[{"id":12,"name":"Adaptateur USB-C","price":19.9},{"id":4,"name":"Câble HDMI 2 m","price":9.5}, …]
```

Evidence 2. False condition: the same products are sorted by price

```
GET /v2/products?sort=(CASE+WHEN+(SELECT+substr(current_user,1,1))='x'+THEN+name+ELSE+price+END) HTTP/1.1
Host: api.example.com

HTTP/1.1 200 OK

[{"id":4,"name":"Câble HDMI 2 m","price":9.5},{"id":12,"name":"Adaptateur USB-C","price":19.9}, …]
```

STEPS TO REPRODUCE

1. Send `GET /v2/products?sort=name` and note the order of the results.
2. Send `GET /v2/products?sort=(CASE WHEN (1=1) THEN name ELSE price END)`: the order is the same.
3. Replace `(1=1)` with `(1=2)`: the results are sorted by price, which confirms that the database evaluates the injected expression.
4. Replace the condition with `(SELECT substr(current_user,1,1))='s'` to read data character by character.

RECOMMENDATION

Never insert a user-supplied value into an SQL query. For sorting, accept only a closed list of values (`name`, `price`, `created_at`) mapped on the server to fixed column names, and use parameterised queries for every other value.

Reduce the privileges of the `shop_app` account to the strict minimum, then review the API access logs for signs of earlier exploitation.

FURTHER READING

- OWASP — SQL Injection Prevention Cheat Sheet
  https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- MITRE — CWE-89: Improper Neutralization of Special Elements used in an SQL Command
  https://cwe.mitre.org/data/definitions/89.html

VUL-02

Access to other customers' invoices by changing the identifier

Severity	HIGH **7.1** / 10 (CVSS 4.0)
CVSS 4.0 vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`
Affected asset	`https://api.example.com/v2/invoices/{id}`
References	CWE-639 · OWASP WSTG-ATHZ-04 · OWASP API1:2023

DESCRIPTION

The `GET /v2/invoices/{id}` endpoint returns the requested invoice without checking that it belongs to the authenticated user's account. As identifiers are sequential, every invoice can be enumerated.

To avoid accessing real customers' data, the proof was established between the two test accounts provided: the `client-a` account could read an invoice of the `client-b` account.

IMPACT

Any customer with an account can read the invoices of every other customer: names, billing addresses, amounts and order details. This is personal data: exploitation would be a personal data breach under the GDPR.

EVIDENCE

Evidence 1. The client-a token gives access to an invoice of the client-b account (masked excerpt)

```
GET /v2/invoices/10482 HTTP/1.1
Host: api.example.com
Authorization: Bearer eyJhbGciOi…(client-a)

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

{"id":10482,"account":"client-b","customer":"T*** C***","total":"249.00","lines":[…]}
```

STEPS TO REPRODUCE

1. Log in with the `client-a` account and retrieve its access token.
2. Log in with the `client-b` account and note the identifier of one of its invoices (here 10482).
3. With the `client-a` token, send `GET /v2/invoices/10482`: the `client-b` invoice is returned.

RECOMMENDATION

Check on the server, for every request, that the requested object belongs to the user's account, preferably in an authorisation component shared by all endpoints. Unpredictable identifiers (UUIDs) make enumeration harder but do not replace this check. Add automated tests covering cross-account access.

FURTHER READING

- OWASP API Security Top 10 2023 — API1: Broken Object Level Authorization
  https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/
- MITRE — CWE-639: Authorization Bypass Through User-Controlled Key
  https://cwe.mitre.org/data/definitions/639.html

VUL-04

No limit on login attempts

Severity	MEDIUM **6.3** / 10 (CVSS 4.0)
CVSS 4.0 vector	`CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N`
Affected asset	`https://api.example.com/v2/session`
References	CWE-307 · OWASP WSTG-ATHN-03

DESCRIPTION

The `POST /v2/session` authentication endpoint accepts an unlimited number of attempts, with no delay, no temporary lockout and no CAPTCHA. The test was limited to 50 consecutive attempts on the `client-a` test account, all processed in under 15 seconds.

IMPACT

An attacker can massively try common or leaked passwords against all accounts (password spraying, credential stuffing). Accounts protected by a weak or reused password can be compromised.

EVIDENCE

Evidence 1. 50 attempts in 14 seconds, no limitation

```
#   heure         statut  réponse
1   10:12:03.114  401     {"error":"invalid_credentials"}
2   10:12:03.402  401     {"error":"invalid_credentials"}
…
50  10:12:17.236  401     {"error":"invalid_credentials"}
```

STEPS TO REPRODUCE

1. Send 50 `POST /v2/session` requests with the `client-a` username and wrong passwords.
2. Observe that every response arrives without delay, account lockout or additional challenge.

RECOMMENDATION

Limit attempts per account and per IP address, with an increasing delay after several failures, without a permanent lockout that could be abused to block accounts. Offer multi-factor authentication, reject passwords known to have leaked, and monitor spikes of failures.

FURTHER READING

- OWASP — Authentication Cheat Sheet
  https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- MITRE — CWE-307: Improper Restriction of Excessive Authentication Attempts
  https://cwe.mitre.org/data/definitions/307.html

VUL-03

Reflected cross-site scripting (XSS) in the search page

Severity	MEDIUM **5.1** / 10 (CVSS 4.0)
CVSS 4.0 vector	`CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:A/VC:N/VI:N/VA:N/SC:L/SI:L/SA:N`
Affected asset	`https://app.example.com/search?q=`
References	CWE-79 · OWASP WSTG-INPV-01

DESCRIPTION

The value of the `q` parameter is written back into the `value` attribute of the search field without encoding. An attacker can close the attribute and inject JavaScript, which runs in the browser of anyone who opens a crafted link. The proof used a harmless payload displaying the domain name. No Content Security Policy (CSP) restricts script execution.

IMPACT

The injected code acts with the victim's session: reading displayed information, performing actions on their behalf, showing a fake login page. The session cookie has the HttpOnly attribute and therefore cannot be read directly by the script.

EVIDENCE

Evidence 1. The payload is written back as-is into the page

```
GET /search?q=%22%3E%3Csvg%20onload%3Dalert(document.domain)%3E HTTP/1.1
Host: app.example.com

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<input type="search" name="q" value="><svg onload=alert(document.domain)>">
```

STEPS TO REPRODUCE

1. Open `https://app.example.com/search?q=%22%3E%3Csvg%20onload%3Dalert(document.domain)%3E` in a browser.
2. Observe a dialog box showing `app.example.com`.

RECOMMENDATION

Encode all data for the context it is inserted into (here, HTML attribute encoding), ideally through the template engine's automatic escaping. Then deploy a strict Content Security Policy as defence in depth.

FURTHER READING

- OWASP — Cross Site Scripting Prevention Cheat Sheet
  https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- MITRE — CWE-79: Improper Neutralization of Input During Web Page Generation
  https://cwe.mitre.org/data/definitions/79.html

VUL-05

Missing Strict-Transport-Security header

Severity	LOW **2.3** / 10 (CVSS 4.0)
CVSS 4.0 vector	`CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:P/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N`
Affected asset	`https://app.example.com`
References	CWE-319 · OWASP WSTG-CONF-07

DESCRIPTION

The application's HTTPS responses do not include the `Strict-Transport-Security` header. The HTTP version (port 80) redirects to HTTPS, but the browser is not told to use HTTPS straight away on later visits.

IMPACT

An attacker on the same network as the victim (public Wi-Fi, for instance) can intercept the first HTTP request and keep the victim on an unencrypted version of the site.

EVIDENCE

Evidence 1. HTTPS response headers

```
HTTP/2 200
content-type: text/html; charset=utf-8
set-cookie: sid=…; Path=/; Secure; HttpOnly; SameSite=Lax
x-content-type-options: nosniff
```

STEPS TO REPRODUCE

1. Run `curl -sI https://app.example.com` and observe that the `Strict-Transport-Security` header is missing.

RECOMMENDATION

Add `Strict-Transport-Security: max-age=31536000; includeSubDomains` after checking that every subdomain is served over HTTPS, then consider enrolling in the HSTS preload list.

FURTHER READING

- OWASP — HTTP Strict Transport Security Cheat Sheet
  https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
- RFC 6797 — HTTP Strict Transport Security (HSTS)
  https://www.rfc-editor.org/rfc/rfc6797

VUL-06

Disclosure of server component versions

Severity	INFO
CVSS 4.0 vector	Not applicable (informational finding)
Affected asset	`https://app.example.com, api.example.com`
References	CWE-497 · OWASP WSTG-INFO-02

DESCRIPTION

Responses include the `Server: nginx/1.24.0` and `X-Powered-By: Express` headers. No exploitable vulnerability related to these versions was identified during the tests.

IMPACT

This information helps an attacker target known vulnerabilities when new flaws are published.

EVIDENCE

Evidence 1. Response headers

```
HTTP/1.1 200 OK
Server: nginx/1.24.0
X-Powered-By: Express
```

STEPS TO REPRODUCE

1. Run `curl -sI https://api.example.com/v2/products` and read the `Server` and `X-Powered-By` headers.

RECOMMENDATION

Hide versions: `server_tokens off;` in the nginx configuration, and `app.disable('x-powered-by')` in the Express application.

FURTHER READING

- OWASP WSTG — Fingerprint Web Server
  https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server

5 Remediation plan

The plan below orders the fixes by vulnerability severity. Timeframes are indicative; adjust them to the exposure and business criticality of the systems concerned.

ID	RECOMMENDED ACTION	SEVERITY
Immediate — within 7 days		
VUL-01	Closed list of sort columns, parameterised queries, log review SQL injection in the sort parameter of the products API	CRITICAL
Short term — within 30 days		
VUL-02	Server-side ownership check on every object Access to other customers' invoices by changing the identifier	HIGH
Medium term — within 90 days		
VUL-04	Progressive attempt limiting, MFA No limit on login attempts	MEDIUM
VUL-03	Context-aware output encoding, then CSP Reflected cross-site scripting (XSS) in the search page	MEDIUM
With the maintenance cycle		
VUL-05	Add the HSTS header Missing Strict-Transport-Security header	LOW
VUL-06	Hide version headers Disclosure of server component versions	INFO

Once the fixes are deployed, a targeted retest confirms that they are effective.

A Appendix — Timeline

DATE AND TIME	EVENT
27 Nov 2026 11:15	Scope v2 frozen, authorisation letter AUT-2026-0001 issued
30 Nov 2026 10:40	Authorisation letter signed by the Client and checked
8 Dec 2026 09:02	Tests start from 198.51.100.7, after the expert's confirmation
8 Dec 2026 14:37	VUL-01 identified; exploitation limited to the proof
8 Dec 2026 14:58	Alert sent to the Client (critical vulnerability)
10 Dec 2026 17:41	End of tests, network filter closed
14 Dec 2026 16:00	Draft report
17 Dec 2026 11:00	Expert review completed, report delivered

Times are given in Brussels time.

B Appendix — Tools

Categories of tools used during the engagement. All test traffic went through the scope enforcement described in section 2.4.

TOOL CATEGORY	USE
Intercepting proxy	Capture, replay and logging of HTTP(S) requests
Automated scanners	Discovery of exposed services and content, detection of known weaknesses, rate-limited
Custom scripts	Targeted checks and non-destructive proofs of concept
AI-based analysis tools	Analysis of responses and help with triage, under the expert's review
Web browser and developer tools	Manual verification and reproduction of vulnerabilities
CVSS 4.0 calculator	Severity scoring according to the FIRST specification

C Appendix — Glossary

ASVS
Application Security Verification Standard: application security requirements published by OWASP.

Black box, grey box
Test performed without an account (black box) or with test accounts provided by the Client (grey box).

CVSS
Common Vulnerability Scoring System: vulnerability severity rating system published by FIRST (version 4.0).

CWE
Common Weakness Enumeration: catalogue of software weakness types maintained by MITRE.

HSTS
HTTP Strict Transport Security: header that makes the browser use HTTPS.

IDOR
Insecure Direct Object Reference: access to another user's object by changing its identifier, for lack of an authorisation check.

Non-destructive exploitation
Demonstrating a vulnerability without altering the data or the availability of the system.

OWASP
Open Worldwide Application Security Project: foundation that publishes application security frameworks.

PTES
Penetration Testing Execution Standard: reference description of how a penetration test is conducted.

Scope
List of authorised targets, set in the authorisation letter and identified by its SHA-256 fingerprint.

Source IP
IP address the test traffic comes from, declared in the authorisation letter.

SQL injection
Insertion of SQL code into a database query through user-supplied data.

Test window
Period, set in the authorisation letter, during which testing is authorised.

WSTG
Web Security Testing Guide: web application security testing guide published by OWASP.

XSS
Cross-site scripting: execution of injected JavaScript code in a user's browser.