Resync

Example App Penetration Test Report

Security Assessment · Confidential

ORGANISATION
Example BV

TEST PERIOD
7 – 11 april 2026

REPORT DATE
21 april 2026

PERFORMED BY
Sofyan Aarrass – Resync

VERSION
1.0

Verify the authenticity of this report at re-sync.nl/verify · Report ID: RSY-2026-0421-VBBV

Contents

1 Introduction

This document describes the results of a penetration test carried out on the client's web application. The purpose of this report is to provide insight into security risks and to support decision-making and risk reduction.

The content of this report is divided into:

- A management summary with a concise overview and general recommendations
- Detailed technical findings
- Concrete recommendations to mitigate the risks

1.1 Disclaimer

This report is confidential and intended solely for the client. Its contents may not be shared with third parties without the client's written consent.

The penetration test was carried out within an agreed scope and time frame. Vulnerabilities outside this scope may not have been identified.

The findings in this report represent a snapshot based on the test period (7 – 11 april 2026) and provide no guarantee that systems are entirely free of security risks.

1.2 Scope and Context

The penetration test was performed as a grey-box web application test: testing was carried out from the perspective of an authenticated user with pre-supplied test accounts, supplemented by research from the perspective of an external, unauthenticated attacker.

The following components were in scope:

- The public web application at `https://app.voorbeeldbv.nl`
- The underlying REST API (`/api/v1/*`)
- Publicly available information about Example BV (OSINT)

The following were explicitly out of scope:

- Denial-of-Service (DoS/DDoS) attacks
- Social engineering targeting Example BV employees
- Physical security of office locations
- Third-party infrastructure (including hosting and CDN providers)

1.3 Approach and Methodology

The assessment followed a fixed, repeatable methodology based on the *OWASP Web Security Testing Guide* (WSTG) and the *OWASP Application Security Verification Standard* (ASVS). The test consisted of the following phases:

1. Reconnaissance – mapping the application, its functionality and attack surface
2. Automated testing – using tooling to quickly rule out known classes of vulnerabilities
3. Manual analysis – in-depth, manual investigation of logic, authorisation and application-specific vulnerabilities that tooling does not detect
4. Validation – every finding is manually confirmed with a working proof of concept before it is included in this report
5. Reporting – documenting findings, risks and concrete recommendations

All findings in this report have been manually validated; no unverified scanner results are reported.

1.4 CVSS and Risk Rating

1.4.1 CVSS score

To determine the severity of vulnerabilities we use, among other things, the *Common Vulnerability Scoring System* (CVSS v3.1).

The CVSS score is based on factors such as:

- Attack complexity
- Required privileges
- User interaction
- Impact on confidentiality, integrity and availability

The score ranges from 0.0 (low risk) to 10.0 (critical risk).

1.4.2 Risk levels

The table below describes the classification of findings based on the CVSS score.

Level	CVSS	Description
Critical	9.0 – 10.0	Directly exploitable with high impact. Requires immediate attention.
High	7.0 – 8.9	Significant risk with potential impact on systems or data.
Medium	4.0 – 6.9	Limited impact or non-trivial to exploit.
Low	0.1 – 3.9	Low impact or difficult to exploit.
Informational	0.0	No direct risk, but relevant to security awareness.

1.5 Contact Information

For questions about this report or its findings, please contact:

Name:	Sofyan Aarrass
Organisation:	Resync
Email:	info@re-sync.nl
Phone:	+31 6 87820340

2 Management Summary

On behalf of Example BV, Resync carried out a penetration test on Example App during 7 – 11 april 2026. The aim of the assessment was to determine to what extent the application withstands attacks that could affect the confidentiality, integrity or availability of data.

The assessment identified several findings, ranging from critical to informational. The most urgent finding (WEB-01, SQL Injection) allows an attacker without any prior knowledge to bypass authentication and read or modify the underlying database. This finding should take priority over the others. The table below shows the distribution of findings by risk level.

No indications were found that the identified vulnerabilities have already been exploited.

2.1 Vulnerability Overview

Risk Level	Count
Critical	1
High	8
Medium	6
Low	1
Informational	1

2.1.1 List of Findings

ID	Title	Risk	CVSS
WEB-01	SQL Injection (SQLi)	Critical	9.8
WEB-02	Insecure Direct Object Reference (IDOR)	High	8.1
WEB-03	No Protection Against Brute-Force on the Login Page	High	7.5
WEB-04	Local File Inclusion (LFI)	High	7.5
WEB-05	Server-Side Request Forgery (SSRF)	High	7.1
WEB-06	HTTP Request Smuggling	High	8.7
WEB-07	Insecure CORS Configuration	High	7.4
WEB-08	Web Cache Poisoning	High	7.5
WEB-09	XML External Entity (XXE) Injection	High	7.5
WEB-10	Host Header Injection and Confusion	Medium	6.5
WEB-11	Open Redirect	Medium	4.3
WEB-12	Cross-Site Request Forgery (CSRF)	Medium	6.5
WEB-13	Reflected Cross-Site Scripting (XSS)	Medium	6.1
WEB-14	Outdated Software with Known Vulnerabilities	Medium	4.2
WEB-15	Informative Error Messages (Stack Traces)	Medium	5.3
WEB-16	Insecure Session Cookies (Missing Attributes)	Low	3.1
WEB-17	Missing Security Headers	Informational	-

2.2 Conclusion

The overall risk posture of Example App requires attention in the short term: one finding was rated critical, alongside several findings rated high. The identified issues are specific and can be resolved within a well-defined development effort; there are no indications of structural shortcomings in the design of the application.

We recommend resolving the critical and high-risk findings as a priority and addressing the remaining findings as part of the regular maintenance process. Once the recommended measures have been implemented, we advise a targeted retest so that it can be established with certainty that the risks have been removed. See chapter 4 for the recommended next steps.

3 Technical Findings

3.1 WEB-01 – SQL Injection (SQLi)

Critical

LOCATION	POST /login (parameter username)
STATUS	Open
RISK LEVEL	Critical
CVSS SCORE	9.8 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H[1])

Description: The login functionality inserts the value of the username parameter directly into an SQL query, without using parameterised statements. This allows an attacker to influence the structure of the underlying query.

Impact:
An attacker can bypass authentication and log in without valid credentials. In addition, the entire database can be read, modified or deleted. Depending on the privileges of the database account, execution of system commands may also be possible in some cases.

Proof of Concept:
By placing a single-quote payload in the username field, the password part of the query is commented out and the login succeeds without a valid password:

Listing 1: Code fragment (http)

```
POST /login HTTP/1.1
Host: vulnerable.example.com
Content-Type: application/x-www-form-urlencoded

username=admin'--&password=arbitrary
```

The application responds with a valid session for the admin account. The vulnerability was additionally confirmed in an automated manner, retrieving the names of all database tables.

Recommendation:
Use parameterised queries (prepared statements) exclusively, or an ORM that binds parameters safely. Never execute user input together with the query text. In addition, grant the database account only the privileges that are strictly necessary.

References: OWASP A03:2021 – Injection[2], CWE-89 – SQL Injection[3]

[2] https://owasp.org/Top10/A03_2021-Injection/
[3] https://cwe.mitre.org/data/definitions/89.html

3.2 WEB-02 – Insecure Direct Object Reference (IDOR)

High

LOCATION	GET /api/v1/users/{id}
STATUS	Open
RISK LEVEL	High
CVSS SCORE	8.1 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N[4])

Description: The API returns account data based on an incrementing numeric `id`, without checking whether the account belongs to the logged-in user. By incrementing or decrementing this number, the data of other users becomes accessible. The associated `PUT` request also allows that data to be modified.

Impact:
An authenticated user can view and modify the personal data of all other users, including names, email addresses and phone numbers. This affects the confidentiality and integrity of customer data and poses a risk under the GDPR.

Proof of Concept:
The logged-in user has `id` 1043. By changing this number, the data of another account becomes visible:

Listing 2: Code fragment (http)

```
GET /api/v1/users/1044 HTTP/1.1
Host: vulnerable.example.com
Authorization: Bearer <own-session-token>

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

{"id":1044,"name":"J. de Vries","email":"j.devries@example.com"}
```

Recommendation:
On every request, verify server-side that the requested object actually belongs to the logged-in user. Base this check on the session rather than on a value taken from the request. In addition, consider using non-guessable identifiers (UUIDs).

References: OWASP A01:2021 – Broken Access Control[5], CWE-639 – Authorization Bypass Through User-Controlled Key[6]

[5] https://owasp.org/Top10/A01_2021-Broken_Access_Control/
[6] https://cwe.mitre.org/data/definitions/639.html

3.3 WEB-03 – No Protection Against Brute-Force on the Login Page

High

LOCATION	POST /login
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N[7])

Description: The login page has no form of delay, account lockout or CAPTCHA after repeated failed login attempts. There is also no limit on the number of requests per IP address or per account.

Impact:
An attacker can automatically try large numbers of passwords or reuse credentials leaked from other services. This significantly increases the likelihood of account takeover, especially for users with a weak password.

Proof of Concept:
More than a thousand consecutive login attempts with different passwords for the same account were sent:

Listing 3: Code fragment (http)

```
POST /login HTTP/1.1
Host: vulnerable.example.com
Content-Type: application/x-www-form-urlencoded

username=victim&password=Summer2024
```

The application kept processing every request normally. No lockout, delay or block occurred, and no record of the attempts was logged.

Recommendation:
Limit the number of login attempts per account and per IP address, and add an increasing delay after consecutive failures. Consider multi-factor authentication and ensure that suspicious patterns are logged and monitored.

References: OWASP A07:2021 – Identification and Authentication Failures[8], CWE-307 – Improper Restriction of Excessive Authentication Attempts[9]

[8] https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
[9] https://cwe.mitre.org/data/definitions/307.html

3.4 WEB-04 – Local File Inclusion (LFI)

High

LOCATION	GET /index.php (parameter page)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N[10])

Description: The page parameter is used directly to include files without any restrictions.

Impact:
An attacker can read sensitive files, including configuration files and source code.

Proof of Concept:
By applying directory traversal, the file /etc/passwd can be read.

Listing 4: Code fragment (http)

```
GET /index.php?page=../../../../etc/passwd HTTP/1.1
Host: vulnerable.example.com
```

Recommendation:
Use an allowlist of permitted files and prevent directory traversal by not using user input directly for file access.

References: OWASP A01:2021 – Broken Access Control[11], CWE-22 – Path Traversal[12]

[11] https://owasp.org/Top10/A01_2021-Broken_Access_Control/
[12] https://cwe.mitre.org/data/definitions/22.html

3.5 WEB-05 – Server-Side Request Forgery (SSRF)

High

LOCATION	POST `/api/v1/import` (parameter `url`)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.1 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:N[13])

Description: The import function fetches a user-supplied URL on the server side in order to process its content. The destination of this request is not validated, which means internal addresses can also be reached.

Impact:
An attacker can make the server issue requests to internal systems that are normally unreachable, or to the cloud environment's metadata endpoint. This allows internal services to be mapped and, in some environments, temporary credentials to be captured.

Proof of Concept:
By supplying the cloud environment's metadata address as the `url`, the server returns its content:

Listing 5: Code fragment (http)

```
POST /api/v1/import HTTP/1.1
Host: vulnerable.example.com
Content-Type: application/json

{"url": "http://169.254.169.254/latest/meta-data/"}
```

Recommendation:
Only allow requests to a predefined list of trusted domains. Block internal IP ranges and the metadata address, and do not follow automatic redirects to unexpected destinations.

References: OWASP A10:2021 – Server-Side Request Forgery (SSRF)[14], CWE-918 – Server-Side Request Forgery[15]

[14] https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/
[15] https://cwe.mitre.org/data/definitions/918.html

3.6 WEB-06 – HTTP Request Smuggling

High

LOCATION	Global (front-end proxy and back-end server)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	8.7 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:N[16])

Description: The front-end proxy and the back-end server determine the length of an HTTP request differently when both a `Content-Length` and a `Transfer-Encoding` header are present. As a result, part of a request may be treated as complete by one system while the other interprets it as the start of a subsequent request.

Impact:
An attacker can send a hidden request that is attached to the next user on the same connection. This can be used to bypass access controls, hijack other users' requests and manipulate responses.

Proof of Concept:
The following request contains conflicting length indicators, which the two systems process differently:

Listing 6: Code fragment (http)

```
POST / HTTP/1.1
Host: app.voorbeeldbv.nl
Content-Length: 6
Transfer-Encoding: chunked

0

G
```

The leftover `G` is prepended to the next request on the connection, demonstrating the desynchronisation between the two systems.

Recommendation:
Ensure that the front-end and the back-end process requests identically. Reject requests that contain both a `Content-Length` and a `Transfer-Encoding` header, and preferably use HTTP/2 up to the back-end server.

References: OWASP A05:2021 – Security Misconfiguration[17], CWE-444 – Inconsistent Interpretation of HTTP Requests[18]

[17] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[18] https://cwe.mitre.org/data/definitions/444.html

3.7 WEB-07 – Insecure CORS Configuration

High

LOCATION	`Access-Control-Allow-Origin` (API)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.4 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:N/A:N[19])

Description: The API reflects the value of the `Origin` header unchanged into the `Access-Control-Allow-Origin` header and sends `Access-Control-Allow-Credentials: true` along with it. As a result, every origin is treated as trusted, including domains controlled by an attacker.

Impact:
When a logged-in victim visits a malicious website, that website can make authenticated requests to the API on the victim's behalf and read the responses. In this way, sensitive user data can be captured.

Proof of Concept:
The API accepts an arbitrary origin and allows credentials at the same time:

Listing 7: Code fragment (http)

```
GET /api/v1/account HTTP/1.1
Host: app.voorbeeldbv.nl
Origin: https://attacker.example

HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://attacker.example
Access-Control-Allow-Credentials: true
```

Recommendation:
Only allow origins that appear in a predefined list of trusted domains. Do not reflect the `Origin` header unchanged, and never combine a broad origin with `Access-Control-Allow-Credentials: true`.

References: OWASP A05:2021 – Security Misconfiguration[20], CWE-942 – Permissive Cross-domain Policy with Untrusted Domains[21]

[20] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[21] https://cwe.mitre.org/data/definitions/942.html

3.8 WEB-08 – Web Cache Poisoning

High

LOCATION	Global (shared cache, `X-Forwarded-Host`)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.5 (CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:N[22])

Description: The application stores responses in a shared cache, but in doing so carries the value of the `X-Forwarded-Host` header over into the content of the response without making that header part of the cache key. As a result, a response influenced by an attacker is stored and subsequently served to other users.

Impact:
An attacker can poison the cache for a frequently requested page, so that all subsequent visitors of that page receive content determined by the attacker. In practice this leads to the loading of a malicious script, and therefore to the execution of JavaScript in the browser of a large number of users.

Proof of Concept:
The `X-Forwarded-Host` is reflected in a script source and the response is cached, as shown by the `X-Cache` header:

Listing 8: Code fragment (http)

```
GET /?cb=12345 HTTP/1.1
Host: app.voorbeeldbv.nl
X-Forwarded-Host: attacker.example

HTTP/1.1 200 OK
X-Cache: miss

<script src="https://attacker.example/main.js"></script>
```

A subsequent, regular request for the same page receives the same poisoned response from the cache (`X-Cache: hit`).

Recommendation:
Include all input that influences the content of a response in the cache key, or exclude such responses from caching. Do not process untrusted headers such as `X-Forwarded-Host` in the content of a response.

References: OWASP A05:2021 – Security Misconfiguration[23], CWE-349 – Acceptance of Extraneous Untrusted Data With Trusted Data[24]

[23] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[24] https://cwe.mitre.org/data/definitions/349.html

3.9 WEB-09 – XML External Entity (XXE) Injection

High

LOCATION	POST /soap/orders (Content-Type: application/xml)
STATUS	Open
RISK LEVEL	High
CVSS SCORE	7.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N[25])

Description: The application processes user-supplied XML with a parser that permits external entities and DOCTYPE declarations. This allows an attacker to define an entity that references a local file or an internal URL.

Impact:
An attacker can read sensitive files from the server, including configuration files containing credentials. In addition, the parser can be abused to make requests to internal systems (server-side request forgery) or to affect the availability of the service.

Proof of Concept:
The following XML defines an external entity that reads the file /etc/passwd and returns it in the response:

Listing 9: Code fragment (xml)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<order><customer>&xxe;</customer></order>
```

The content of the file is processed and displayed in the application's response.

Recommendation:
Fully disable the processing of external entities and DOCTYPE declarations in the XML parser used. Preferably use an up-to-date parser with a secure default configuration, and do not process untrusted XML with the default settings.

References: OWASP A05:2021 – Security Misconfiguration[26], CWE-611 – Improper Restriction of XML External Entity Reference[27]

[26] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[27] https://cwe.mitre.org/data/definitions/611.html

3.10 WEB-10 – Host Header Injection and Confusion

Medium

LOCATION	Global (`Host` and `X-Forwarded-Host` headers)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	6.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N[28])

Description: The application trusts the value of the `Host` header and the `X-Forwarded-Host` header when constructing absolute URLs, among other things in password-reset emails. An attacker can modify these headers at will. In addition, the server accepts the absolute (proxy) form of the request line, in which the host in the request line differs from the `Host` header.

Impact:
By manipulating the host, an attacker can make the link in a password-reset email point to their own domain. If the victim clicks that link, the reset token is sent to the attacker, which can lead to account takeover. The differing interpretation of the host can additionally be used to bypass access rules that are based on the host.

Proof of Concept:
The server accepts a differing host in the request line (proxy form), while the `Host` header indicates a different value:

Listing 10: Code fragment (http)

```
GET https://internal.voorbeeldbv.nl/ HTTP/1.1
Host: app.voorbeeldbv.nl
```

On a password-reset request, the `X-Forwarded-Host` is carried over into the link in the email:

Listing 11: Code fragment (http)

```
POST /password-reset HTTP/1.1
Host: app.voorbeeldbv.nl
X-Forwarded-Host: attacker.example
Content-Type: application/json

{"email": "victim@example.com"}
```

The email that is sent then contains a reset link to `https://attacker.example/reset?token=....`

Recommendation:
Determine the host name server-side based on a fixed, configured value instead of the headers supplied by the client. Ignore `X-Forwarded-Host` unless it originates from a trusted proxy, and reject requests whose host does not match an allowlist.

References: OWASP A05:2021 – Security Misconfiguration[29], CWE-20 – Improper Input Validation[30]

[29] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[30] https://cwe.mitre.org/data/definitions/20.html

3.11 WEB-11 – Open Redirect

Medium

LOCATION	`GET /redirect` (parameter `next`)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	4.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N[31])

Description: The `next` parameter determines the URL to which the user is redirected after an action, without the supplied value being validated. This allows redirection to an arbitrary external domain.

Impact:
An attacker can craft a trustworthy-looking link on the organisation's domain that leads the victim to a malicious website. This increases the credibility of phishing and can be used to harvest credentials.

Proof of Concept:
The following link on the trusted domain redirects the user to an external site:

Listing 12: Code fragment (http)

```
GET /redirect?next=https://attacker.example/login HTTP/1.1
Host: app.voorbeeldbv.nl

HTTP/1.1 302 Found
Location: https://attacker.example/login
```

Recommendation:
Only allow redirects to internal, relative paths or to a fixed list of permitted destinations. Reject absolute URLs to external domains.

References: OWASP A01:2021 – Broken Access Control[32], CWE-601 – URL Redirection to Untrusted Site[33]

[32] https://owasp.org/Top10/A01_2021-Broken_Access_Control/
[33] https://cwe.mitre.org/data/definitions/601.html

3.12 WEB-12 – Cross-Site Request Forgery (CSRF)

Medium

LOCATION	`POST /account/email` (change email address)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	6.5 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N[34])

Description: State-changing actions, such as changing an account's email address, are accepted solely on the basis of the session cookie. No anti-CSRF token is used, and the session cookie has no restrictive `SameSite` setting. As a result, such a request can also be triggered from an external, malicious website.

Impact:
When a logged-in victim visits a page set up by the attacker, a sensitive action is performed on the victim's behalf without their knowledge. By changing the account's email address, the attacker can then initiate a password reset and thus take over control of the account.

Proof of Concept:
The following page automatically submits a request as soon as a logged-in victim visits it:

Listing 13: Code fragment (html)

```
<form action="https://app.voorbeeldbv.nl/account/email" method="POST">
  <input type="hidden" name="email" value="attacker@example.com">
</form>
<script>document.forms[0].submit();</script>
```

The request is processed by the application and the account's email address is changed.

Recommendation:
Protect state-changing requests with an anti-CSRF token that is unique per session and validated server-side. In addition, set session cookies with `SameSite=Lax` or `Strict` and, where possible, verify the `Origin` or `Referer` header.

References: OWASP A01:2021 – Broken Access Control[35], CWE-352 – Cross-Site Request Forgery[36]

[35] https://owasp.org/Top10/A01_2021-Broken_Access_Control/
[36] https://cwe.mitre.org/data/definitions/352.html

3.13 WEB-13 – Reflected Cross-Site Scripting (XSS)

Medium

LOCATION	`GET /search` (parameter `q`)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	6.1 (CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N[37])

Description: The search functionality processes user input without context-aware output encoding.

Impact:
An attacker can execute JavaScript in the context of the user, which can lead to session hijacking.

Proof of Concept:
The following request demonstrates the vulnerability:

Listing 14: Code fragment (http)

```
GET /search?q=<script>alert(1)</script> HTTP/1.1
Host: vulnerable.example.com
```

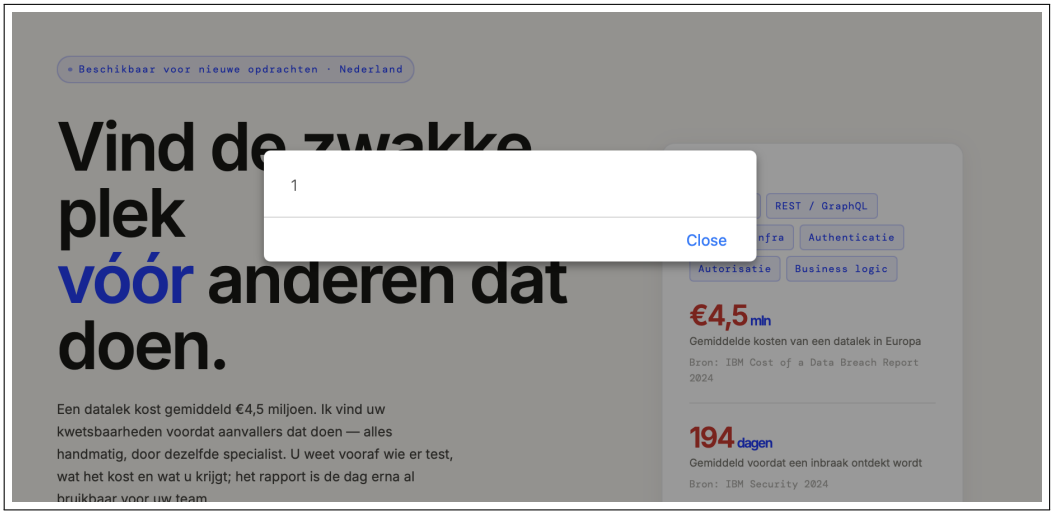

Figure 1: The payload is returned unprocessed and executed as JavaScript in the browser

Recommendation:
Implement context-aware output encoding and validate all user input.

References: OWASP A03:2021 – Injection[38], CWE-79 – Cross-site Scripting[39]

[38] https://owasp.org/Top10/A03_2021-Injection/
[39] https://cwe.mitre.org/data/definitions/79.html

3.14 WEB-14 – Outdated Software with Known Vulnerabilities

Medium

LOCATION	Frontend (`jQuery 1.8.3`)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	4.2 (CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N[40])

Description: The application loads the JavaScript library jQuery in version 1.8.3. Several publicly known vulnerabilities have been registered for this version, including cross-site scripting through unsafe processing of input. This version has not been supported for years.

Impact:
Because this is a publicly known version, the associated vulnerabilities and exploit code are easy to find. An attacker can use these, for example, to execute script code in a user's browser.

Proof of Concept:
The version can be derived from the loaded source in the HTML:

Listing 15: Code fragment (html)

```
<script src="/assets/js/jquery-1.8.3.min.js"></script>
```

Among others, CVE-2015-9251 and CVE-2019-11358 apply to this version.

Recommendation:
Update the library to the most recent supported version and then test the application for regressions. Maintain an up-to-date overview of the dependencies used and check it periodically and automatically for known vulnerabilities.

References: OWASP A06:2021 – Vulnerable and Outdated Components[41], CWE-1104 – Use of Unmaintained Third Party Components[42]

[41] https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/
[42] https://cwe.mitre.org/data/definitions/1104.html

3.15 WEB-15 – Informative Error Messages (Stack Traces)

Medium

LOCATION	Application-wide (e.g. `/api/v1/*`)
STATUS	Open
RISK LEVEL	Medium
CVSS SCORE	5.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N[43])

Description: On unexpected server errors, the application returns a full stack trace, including internal file paths and the framework version in use.

Impact:
An attacker can use this information to infer the technology stack and search specifically for known vulnerabilities, which facilitates follow-up attacks.

Proof of Concept:
Sending an invalid parameter leads to an unhandled error and a full stack trace in the response:

Listing 16: Code fragment (http)

```
GET /api/v1/orders?id=invalid HTTP/1.1
Host: vulnerable.example.com

HTTP/1.1 500 Internal Server Error
Content-Type: text/plain

TypeError: Cannot read property 'id' of undefined
    at OrdersController.show (/srv/app/controllers/orders.js:42)
    at Layer.handle (/srv/app/node_modules/express/lib/router/layer.js:95)
```

Recommendation:
Disable verbose error reporting in production and show users a generic error message. Log the full details server-side for diagnosis.

References: OWASP A05:2021 – Security Misconfiguration[44], CWE-209 – Generation of Error Message Containing Sensitive Information[45]

[44] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[45] https://cwe.mitre.org/data/definitions/209.html

3.16 WEB-16 – Insecure Session Cookies (Missing Attributes)

Low

LOCATION	Session cookie `SESSIONID` (global)
STATUS	Open
RISK LEVEL	Low
CVSS SCORE	3.1 (CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N[46])

Description: The session cookie is set by the application without the `Secure`, `HttpOnly` and `SameSite` attributes. As a result, the cookie can be sent over an unencrypted connection and is accessible from JavaScript.

Impact:
The absence of these attributes increases the likelihood that a session cookie is intercepted or read, for example in combination with a cross-site scripting vulnerability. This allows a user's session to be taken over.

Proof of Concept:
The `Set-Cookie` header after login is missing the attributes mentioned:

Listing 17: Code fragment (http)

```
HTTP/1.1 200 OK
Set-Cookie: SESSIONID=8f3b2c9a1d4e7f60; Path=/
```

Recommendation:
Set session cookies with the `Secure`, `HttpOnly` and an appropriate `SameSite` attribute (`Strict` or `Lax`). In addition, limit the validity period of the session.

References: OWASP A05:2021 – Security Misconfiguration[47], CWE-614 – Sensitive Cookie Without Secure Attribute[48]

[47] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[48] https://cwe.mitre.org/data/definitions/614.html

3.17 WEB-17 – Missing Security Headers

Informational

LOCATION	All HTTP responses (global)
STATUS	Open
RISK LEVEL	Informational
CVSS SCORE	- (-)

Description: HTTP responses from the application lack security headers such as `Content-Security-Policy`, `X-Frame-Options` and `Strict-Transport-Security`.

Impact:
The absence of these headers weakens the defence-in-depth against, among other things, clickjacking and content sniffing, and increases the potential impact of other vulnerabilities.

Proof of Concept:

Listing 18: Code fragment (http)

```
$ curl -I https://app.voorbeeldbv.nl/
HTTP/1.1 200 OK
Server: nginx
Content-Type: text/html
```

Headers such as `Content-Security-Policy` and `X-Frame-Options` are missing from the response.

Recommendation:
Implement the missing headers in accordance with the OWASP Secure Headers Project guidelines.

References: OWASP A05:2021 – Security Misconfiguration[49], CWE-693 – Protection Mechanism Failure[50]

[49] https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
[50] https://cwe.mitre.org/data/definitions/693.html

4 Next Steps

Below is a recommended approach for following up on the identified findings.

4.1 Prioritisation

We recommend resolving the findings in the order presented in this report: findings with a higher risk level appear first and deserve priority. If there is any doubt about the right order or approach, you are welcome to contact Resync – see chapter 1.5.

4.2 Retest

Once the recommended measures have been implemented, we offer a targeted retest of the resolved findings. For each finding we confirm whether the measure taken actually removes the risk, so that this can be established with certainty towards internal stakeholders, auditors or customers.

4.3 Remediation Support

Does the development team have questions about implementing a recommendation? We are happy to think along about the chosen solution, whether through a clarification of the finding or a short call with the developers involved.

This report is a snapshot. For continuous insight into the security posture of your applications, we recommend making periodic penetration tests – for example annually or after major changes – part of your development process.