Cyberdefense

Federal Chancellery

Evoting Web Application 1.4.2.1

Security Audit Report

June 2024

Reference	P021810
Classification	PUBLIC
Last modified	2024-07-22

Client contact information

Federal Chancellery

Orange Cyberdefense contact information

Orange Cyberdefense Switzerland
Rue du Sablon 4
1110 Morges
Switzerland

Versions

Date	Version	Author	Description
2024-06-19	0.1	Orange Cyberdefense Switzerland	Initial document
2024-06-19	0.2	Orange Cyberdefense Switzerland	Added summaries
2024-06-24	1.0	Orange Cyberdefense Switzerland	Review

Table of contents

Executive summary

Results summary

Orange Cyberdefense Switzerland was contracted by the Federal Chancellery to assess the security of the E-voting web application developed by Swiss Post. To this end, engineers acted like real attackers and searched for vulnerabilities and weaknesses within the application to determine the risk for the voters and the secrecy and integrity of their votes, but also the robustness of the system during a voting event.

In terms of risks, Orange Cyberdefense Switzerland did not identify any issues that could compromise the confidentiality, integrity or availability of the voting infrastructure which was setup for the test and reachable at `pit.evoting.ch`. Only one informational finding was discovered, in relation to the use of outdated components in certain development dependencies. These are not used in production and thus cannot be exploited, which is why it is noted as an informational finding only.

Indeed, all the application web pages (used by normal voting users and SDM applications) are precompiled using different libraries. Some of these libraries are outdated and contain vulnerable code that may be exploitable if user input reaches specific vulnerable functions. However, no specific exploitable path was found during the timespan of the audit.

From a normal voter standpoint, the attack surface is very limited, as a reverse-proxy filters faulty user inputs. On top of this, the server relies on cryptographic operations to further restrict malicious operations.

The overall risk level is therefore considered as very low.

High level impressions

Strengths
⊕ Small attack surface
⊕ HTTP security headers
⊕ HTTP content secured with cryptographic operations
⊕ Reverse-proxy filter on data

Weaknesses
⊖ Outdated dependencies

Security dashboard

Type	White-box	**Schedule**	2024-06-12 – 2024-06-19
Scope	Evoting application 1.4.2.1	**Effort**	18 days

Global risk level

Attacker profiles	Risk level
1. Without a voting card	
2. With a voting card	
3. Secure Data Manager Context	

Status by attacker profile

OBJECTIVES	1.	2.	3.
Gain access to the internal network	✓	✓	✓
Execute arbitrary commands on a server	✓	✓	✓
Authentication bypass	✓	✓	✓
Vote integrity	✓	✓	✓

✓ NOT COMPROMISED ! PARTIALLY COMPROMISED ! COMPROMISED

Technical summary

Scope

The scope of the audit includes the latest version of the e-voting web application (release `1.4.2.1`), which was instantiated at the following address:

`https://pit.evoting.ch/vote/#/legal-terms/D287064B648D3498176D08978539B9BA`

For the above instance, valid voting cards could be directly generated and downloaded at the following URL:

`https://vcdl.evoting.ch/vc/`

Because of the audited project being open source, the application's source code and development environment, including the documentation, could be found on GitLab:

`https://gitlab.com/groups/swisspost-evoting/`

Restrictions

No social engineering or denial of service attacks were performed during this audit.

Results

On the first day, the engineers began by reading the documentation provided on GitLab to understand the overall infrastructure of the system and its functioning. This step was necessary to deploy the application locally, thanks to the development environment. The aim of having such an instance is to have complete control over the server and to increase the chances of identifying security vulnerabilities in the different parts of the Secure Data Manager, which is used to configure a voting event.

During the building phase, engineers noted potential weaknesses related to the use of outdated dependencies. Although these vulnerabilities require specific conditions that were not present in the test environment, it is recommended to assess the possibility of updating these vulnerable packages and to apply updates where feasible.

Simultaneously, a vulnerability scan was conducted on the application `pit.evoting.ch` hosting the online instance. However, E-Voting System's perimeter includes protective measures such as a Web Application Firewall (WAF) and anti-Denial of Service (DOS) mechanisms, which significantly limits the types of requests that external users can make to the E-Voting server, thereby reducing the attack surface.

Engineers conducted a thorough review of the source code for various components, utilising both manual inspection and automated tools. They also performed multiple tests to identify potential weaknesses, such as logical errors in the application or injection vulnerabilities that could compromise the integrity and privacy of the vote.

Although no concrete risk-related attacks were identified during the audit, a few strange behaviours were discovered and described in the additional remarks section of this report. These include the configuration of the CSP header, which on the first day suffered from a

malformation due to the omission of a simple quote. The value not being considered by the browser can lead to unexpected behaviour and increase the attack surface.

Engineers also discovered a strange redirect to the `demo.evoting.ch` domain when using a malformed URL.

Finally, engineers tried to bypass the character validation allowed when submitting a custom candidate. Nevertheless, it was not possible to inject an unauthorised character into the decrypted result file. Indeed, the character mapping of the decryption process ensures that only characters from the valid set are processed.

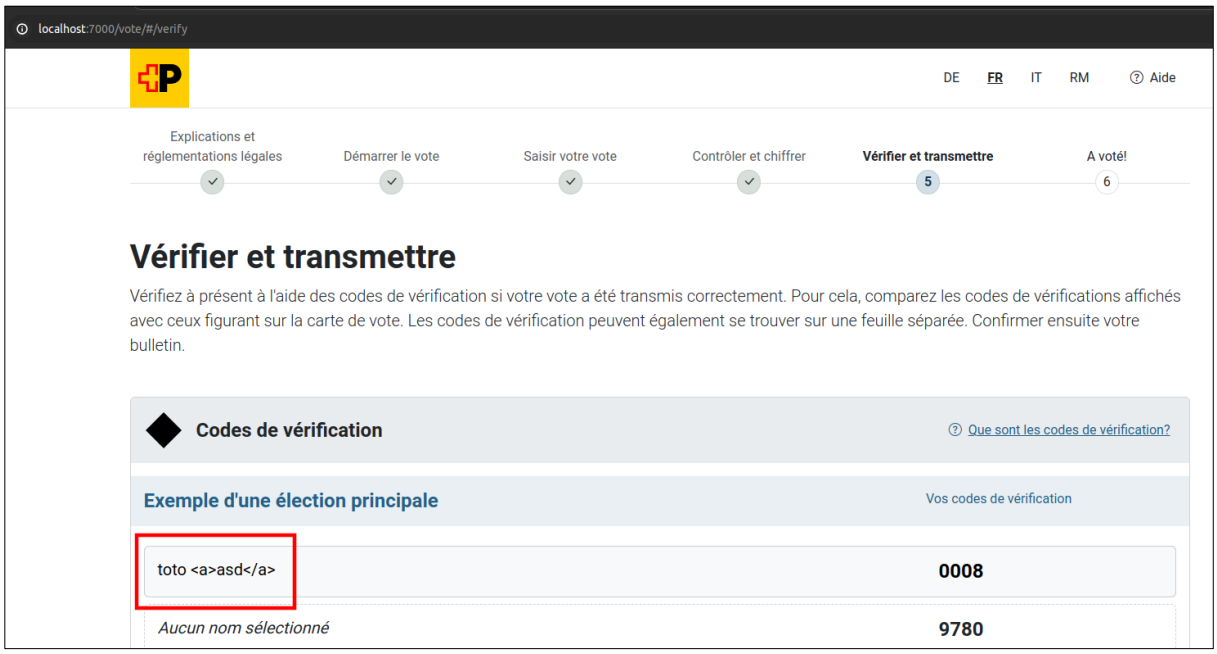
Informational findings

ID	Finding
P021810-01	Outdated software

Additional remarks

Write-in sanitisation

The application supports the so-called `write-ins` to give voters the option of specifying a candidate outside of the predefined lists. The sanitisation of the `write-ins` is done client-side. Thus, the auditors were able to modify the JavaScript code of the page and bypass it. The image below demonstrates the submission of a `write-in` containing unauthorized characters <>:



WriteIn with unauthorised characters.

Nevertheless, it was not possible to inject any unauthorized character inside the decrypted result file. Indeed, the algorithm used to decrypt the `write-in` values only permits authorized characters. As a result, as shown below, no malicious character is produced by the latter.

The decrypted write-in contains only authorized characters.

Redirection to demo instance

Navigating to the URL `https://pit.evoting.ch/vote` results in a 302 HTTP response with the following URL as redirect location:

`https://demo.evoting.ch/vote/`

Broken CSP

During the first day of the audit, the CSP header was malformed, more precisely it was missing a closing single quote:

```
default-src 'none'; script-src 'self' 'wasm-unsafe-eval'; style-src 'self'; img-src 'self' data:;
connect-src 'self'; worker-src 'self'; frame-src 'none'; frame-ancestors 'none'; font-src 'self';
base-uri 'self'; form-action 'none
```

This CSP misconfiguration may cause unpredictable behaviour, depending on the browser used by the users. As pointed out, this issue was observed only during the first day of the audit, the 12th of June 2024.

Detailed results

Informational findings

INFO	P021810-01 Outdated software

PREREQUISITES	COMPROMISED ASSETS
–	Confidentiality and integrity of the JavaScript components

Vulnerable components

- Crypto-primitives-ts
- Secure-data-manager-frontend
- Direct-trust-tool-frontend
- Voter-portal
- Voting-client-js
- Verifier-frontend

Description

An outdated system, or a system using outdated software is more likely to be prone to attacks than a system with all updates and patches installed. A regular update is mandatory in order to correct issues that could enable an attacker to compromise the normal behaviour of the application.

Exploitation

During the build process of the latest e-voting application (version 1.4.2.1), multiple npm warnings related to the use of outdated dependencies affected by known vulnerabilities were observed. By attaching to the docker of a running build and using the command `npm audit` inside the directories of the concerned project, it is possible to retrieve the details of those vulnerabilities. The output of this command for each of the interested projects is shown below.

crypto-primitives-ts

```
~/crypto-primitives-ts$ npm audit
# npm audit report

@babel/traverse  <7.23.2
Severity: critical
Babel vulnerable to arbitrary code execution when compiling specifically crafted malicious code - https://github.com/advisories/GHSA-67hx-6x53-jw92
fix available via `npm audit fix`
node_modules/@babel/traverse

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

2 vulnerabilities (1 high, 1 critical)
```

secure-data-manager-frontend

```
~/e-voting/secure-data-manager/secure-data-manager-frontend$ npm audit
# npm audit report

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

1 high severity vulnerability
```

direct-trust-tool-frontend

```
~/e-voting/tools/direct-trust-tool/direct-trust-tool-frontend$ npm audit
# npm audit report

app-builder-lib  <24.13.2
Severity: high
electron-builder's NSIS installer - execute arbitrary code on the target machine (Windows only) -
https://github.com/advisories/GHSA-r4pf-3v7r-hh55
fix available via `npm audit fix --force`
Will install electron-builder@24.13.3, which is outside the stated dependency range
node_modules/app-builder-lib
  dmg-builder  5.0.0 - 24.13.1
  Depends on vulnerable versions of app-builder-lib
  node_modules/dmg-builder
    electron-builder  19.25.0 || 20.24.0 - 24.13.1
    Depends on vulnerable versions of app-builder-lib
    Depends on vulnerable versions of dmg-builder
    node_modules/electron-builder

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

ejs  <3.1.10
Severity: moderate
ejs lacks certain pollution protection - https://github.com/advisories/GHSA-ghr5-ch3p-vcr6
fix available via `npm audit fix`
node_modules/ejs

express  <4.19.2
Severity: moderate
Express.js Open Redirect in malformed URLs - https://github.com/advisories/GHSA-rv95-896h-c2vc
fix available via `npm audit fix`
node_modules/express

follow-redirects  <=1.15.5
Severity: moderate
follow-redirects' Proxy-Authorization header kept across hosts -
https://github.com/advisories/GHSA-cxjh-pqwp-8mfp
fix available via `npm audit fix`
node_modules/follow-redirects

tar  <6.2.1
Severity: moderate
Denial of service while parsing a tar file due to lack of folders count validation -
https://github.com/advisories/GHSA-f5x3-32g6-xq36
fix available via `npm audit fix`
node_modules/tar

undici  6.0.0 - 6.11.0
```

```
Undici's Proxy-Authorization header not cleared on cross-origin redirect for dispatch, request,
stream, pipeline - https://github.com/advisories/GHSA-m4v8-wqvr-p9f7
Undici's fetch with integrity option is too lax when algorithm is specified but hash value is in
incorrect - https://github.com/advisories/GHSA-9qxr-qj54-h672
fix available via `npm audit fix --force`
Will install @angular-devkit/build-angular@17.3.8, which is outside the stated dependency range
node_modules/undici
  @angular-devkit/build-angular  15.1.0-next.0 - 15.2.10 || 16.0.0-next.0 - 16.2.12 || 17.0.0-
next.0 - 17.3.3 || 18.0.0-next.0 - 18.0.0-rc.3
  Depends on vulnerable versions of undici
  Depends on vulnerable versions of vite
  Depends on vulnerable versions of webpack-dev-middleware
  node_modules/@angular-devkit/build-angular

vite  5.0.0 - 5.0.12
Severity: moderate
Vite's `server.fs.deny` did not deny requests for patterns with directories. -
https://github.com/advisories/GHSA-8jhw-289h-jh2g
fix available via `npm audit fix --force`
Will install @angular-devkit/build-angular@17.3.8, which is outside the stated dependency range
node_modules/vite

webpack-dev-middleware  <=5.3.3 || 6.0.0 - 6.1.1
Severity: high
Path traversal in webpack-dev-middleware - https://github.com/advisories/GHSA-wr3j-pwj9-hqq6
Path traversal in webpack-dev-middleware - https://github.com/advisories/GHSA-wr3j-pwj9-hqq6
fix available via `npm audit fix --force`
Will install @angular-devkit/build-angular@17.3.8, which is outside the stated dependency range
node_modules/webpack-dev-middleware
node_modules/webpack-dev-server/node_modules/webpack-dev-middleware

12 vulnerabilities (1 low, 5 moderate, 6 high)
```

voter-portal

```
~/e-voting/voter-portal$ npm audit
# npm audit report

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

1 high severity vulnerability
```

voting-client-js

```
~/e-voting/voting-client-js$ npm audit
# npm audit report

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

1 high severity vulnerability
```

verifier-frontend

```
~/verifier/verifier-frontend$ npm audit
# npm audit report

braces  <3.0.3
Severity: high
Uncontrolled resource consumption in braces - https://github.com/advisories/GHSA-grv7-fg5c-xmjg
fix available via `npm audit fix`
node_modules/braces

ejs  <3.1.10
Severity: moderate
ejs lacks certain pollution protection - https://github.com/advisories/GHSA-ghr5-ch3p-vcr6
fix available via `npm audit fix`
node_modules/ejs

follow-redirects  <=1.15.5
Severity: moderate
follow-redirects' Proxy-Authorization header kept across hosts -
https://github.com/advisories/GHSA-cxjh-pqwp-8mfp
fix available via `npm audit fix`
node_modules/follow-redirects

tar  <6.2.1
Severity: moderate
Denial of service while parsing a tar file due to lack of folders count validation -
https://github.com/advisories/GHSA-f5x3-32g6-xq36
fix available via `npm audit fix`
node_modules/tar

4 vulnerabilities (3 moderate, 1 high)
```

The libraries highlighted above are outdated and contain vulnerable code that could be exploited if user input reaches certain vulnerable functions. However, no specific exploitable path was identified during the audit period.

For example, in the outdated `@Babel/traverse`, the critical vulnerability may lead to arbitrary code execution, if and only if user input reaches inside the `traverse` function, which was not found in the context of the audit.

Possible solutions

Apply security updates

It is recommended to apply security updates on a regular basis.

Since the outdated components are related to `npm` installation, the versioning files such as `package-lock.json` of each project should be updated.

Complements

Legend

SCRT Score

For each vulnerability discovered and detailed in this report, SCRT provides a threat assessment based on two indicators, an **Impact** and a **Probability** of exploitation.

IMPACT	Impact of the vulnerability in case of successful exploitation ("How bad?")			
N/A	Weak	Medium	High	Critical
PROBABILITY	**Probability that the vulnerability will be discovered and exploited by an attacker?**			
N/A	Low	Medium	High	Very high

However, it is important to keep in mind that this assessment is solely based on the information available to the engineers at the time of the audit. The engineers are not necessarily aware of all the details regarding the vulnerable applications or systems. Consequently, these ratings should always be reconsidered based on the context of the information system as a whole.

CVSS Score

In addition to its own scoring system, SCRT also provides an evaluation based on the **Common Vulnerability Scoring System** (CVSS), for each vulnerability.

As a reminder, CVSS is a vulnerability scoring system designed to provide an open and standardised method for rating IT vulnerabilities. CVSS helps organisations prioritise and coordinate a joint response to security vulnerabilities by communicating the base, temporal and environmental properties of a vulnerability. More information about the CVSS scoring system can be found here: https://www.first.org/cvss/user-guide

Risk calculation

Each risk presented in this report is calculated as the product of an **impact** and a **probability** of exploitation, as defined in the matrix below.

Overall Risk Severity

Impact					
	CRITICAL	High	High	Critical	Critical
	HIGH	Moderate	Moderate	High	Critical
	MODERATE	Low	Moderate	Moderate	High
	LOW	Low	Low	Moderate	High
		LOW	MODERATE	HIGH	CRITICAL

Probability

SCRT provides an estimation of the effort required to fix each vulnerability and thus mitigate their associated risk. It should be noted that this assessment is based on SCRT's experience, and as such might not fully reflect the context of the company or organisation.

Context

The context of each vulnerability is defined by its prerequisites and a list of compromised assets. The prerequisites represent the conditions that are required for the exploitation of a given vulnerability (*e.g.*: social engineering). Compromised assets represent the theoretical or tangible result of its exploitation (*e.g.*: a user account).