securitum

Security report

SUBJECT

Penetration test of Calamari web application

DATE

18.09.2025 – 08.10.2025
27.04.2026 – 30.04.2026 (retest)
03.06.2026 (retest v2)

LOCATION

Cracow (Poland)

AUTHORS

Jacek Siwek
Sebastian Jeż (retests)

VERSION

1.2

securitum

Executive summary

This document is a summary of work conducted by Securitum. The subject of the test was the Calamari web application available at https://calamari.io and its integration with the following services:

- Google Workspace,
- Slack,
- Microsoft 365,
- SAML (Google).

During the work, a tenant provided by the Client was used, which was available at https://[DOMAIN].

The auditor also created his own tenant available at https://[DOMAIN].

Tests were conducted using the following roles:

- Administrator,
- Company manager,
- Team manager,
- Direct manager,
- Unauthenticated user (visitor of the website).

The test was conducted in best effort mode.

The most severe vulnerabilities identified during the assessment were:

- SECURITUM-2418465-001: Authorization – broken access control,
- SECURITUM-2418465-002: Race condition – possibility of exceeding vacation days,
- [REDACTED],
- SECURITUM-2418465-004: Denial of Service.

Information: Due to the specific nature of the environment and the business assumptions, items SECURITUM-2418465-003, SECURITUM-2418465-005, and SECURITUM-2418465-012 have been removed from the public report.

During the tests, particular emphasis was placed on vulnerabilities that might in a negative way affect confidentiality, integrity or availability of processed data.

The security tests were carried out according to generally accepted methodologies, including: OWASP TOP10, (in a selected range) OWASP ASVS 5.0 L2 as well as internal good practices of conducting security tests developed by Securitum.

An approach based on manual tests (using the above-mentioned methodologies), supported by several automatic tools (i.a. Burp Suite Professional, sslscan, ffuf, nmap), was used during the assessment.

The vulnerabilities are described in detail in further parts of the report.

Status after retests

On June 3, 2026, a retest of previously identified vulnerabilities was carried out. The result of the work is the creation of a „Status after retest" section for tested vulnerability, which contains detailed information about the retest performed.

A total of 5 vulnerabilities and 2 informational points were retested, and a summary is provided in the table below:

ID	Severity/Name	Status after retest
SECURITUM-2418465-001	[MEDIUM] Authorization – broken access control	FIXED
SECURITUM-2418465-002	[MEDIUM] Race condition – possibility of exceeding vacation days	FIXED
SECURITUM-2418465-004	[MEDIUM] Denial of Service	FIXED
SECURITUM-2418465-006	[LOW] Support for outdated TLS ciphers	FIXED
SECURITUM-2418465-007	[LOW] Cookies without SameSite attribute set	FIXED
SECURITUM-2418465-008	[LOW] Outdated software	PARTIALLY FIXED
SECURITUM-2418465-009	[LOW] Modifying error messages	FIXED
SECURITUM-2418465-010	[LOW] Redundant information disclosure about the application environment	FIXED
SECURITUM-2418465-011	[LOW] HTML injection	FIXED
SECURITUM-2418465-013	[LOW] Open Redirect – possibility to redirect a user to a malicious domain	FIXED
SECURITUM-2418465-014	[LOW] Basic Authentication	NOT VERIFIED
SECURITUM-2418465-015	[LOW] Accessing the local administrator account without proper permissions	FIXED
SECURITUM-2418465-016	[LOW] Redundant information revealed in detailed error messages	FIXED
SECURITUM-2418465-017	[INFO] Using numerical resource identifiers	NOT VERIFIED
SECURITUM-2418465-018	[INFO] User enumeration	PARTIALLY IMPLEMENTED
SECURITUM-2418465-019	[INFO] X-XSS-Protection header enabled	NOT VERIFIED
SECURITUM-2418465-020	[INFO] Lack of Referrer-Policy header	NOT VERIFIED

SECURITUM-2418465-021	[INFO] Host Header Injection	IMPLEMENETED
SECURITUM-2418465-022	[INFO] Lack of general field validation	IMPLEMENETED
SECURITUM-2418465-023	[INFO] Lack of Content-Disposition header	NOT VERIFIED
SECURITUM-2418465-024	[INFO] Lack of Content-Security-Policy header	PARTIALLY IMPLEMENTED

Risk classification

Vulnerabilities are classified on a five-point scale, that reflects both the probability of exploitation of the vulnerability and the business risk of its exploitation. Below, there is a short description of the meaning of each of the severity levels:

- CRITICAL – exploitation of the vulnerability makes it possible to compromise the server or network device, or makes it possible to access (in read and/or write mode) data with a high degree of confidentiality and significance. The exploitation is usually straightforward, i.e. an attacker does not need to gain access to the systems that are difficult to reach and does not need to perform social engineering. Vulnerabilities marked as 'CRITICAL' must be fixed without delay, mainly if they occur in the production environment.

- HIGH – exploitation of the vulnerability makes it possible to access sensitive data (similar to the 'CRITICAL' level), however the prerequisites for the attack (e.g. possession of a user account in an internal system) make it slightly less likely. Alternatively, the vulnerability is easy to exploit, but the effects are somehow limited.

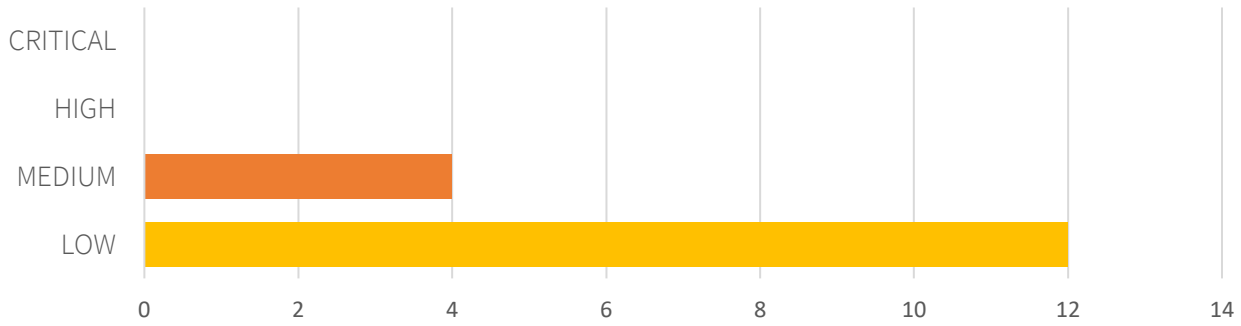
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser degree of significance.

- LOW – exploitation of the vulnerability results in minor direct impact on the security of the test subject or depends on conditions that are very difficult to achieve in practical manner (e.g. physical access to the server).

- INFO – issues marked as 'INFO' are not security vulnerabilities per se. They aim to point out good practices, the implementation of which will lead to the overall increase of the system security level. Alternatively, the issues point out some solutions in the system (e.g. from an architectural perspective) that might limit the negative effects of other vulnerabilities.

Statistical overview

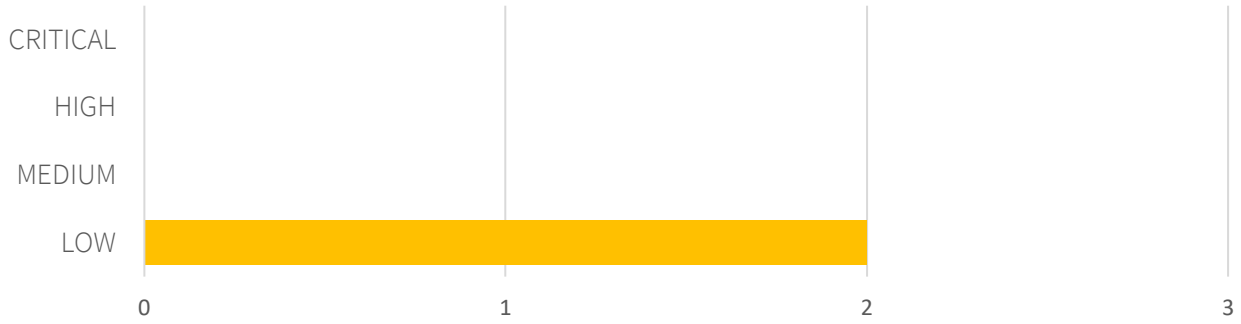
Below, a statistical summary of vulnerabilities is shown:



Additionally, 8 INFO issues were reported.

Statistical overview after retests

Below, a statistical summary of vulnerabilities is shown:



Additionally, 6 INFO issues remain to be reported.

Contents

Change history

Document date	Version	Change description
03.06.2026	1.2	Creation of the security report after the retest. The following changes have been added: • „Status after retest" section for each item, • „Status after retest" section in the summary, • "Statistical overview after retest" chart.
30.04.2026	1.1	Creation of the security report after the retest. The following changes have been added: • „Status after retest" section for each item, • „Status after retest" section in the summary, • "Statistical overview after retest" chart.
08.10.2025	1.0	Creation of a security report.

Vulnerabilities in the web application

[FIXED][MEDIUM] SECURITUM-2418465-001: Authorization – broken access control

STATUS AFTER RETEST (03.06.2026)

The vulnerability has been fixed.

The Client indicated that the behaviour described in examples #1 and #2 is intended and should not be considered broken access control.

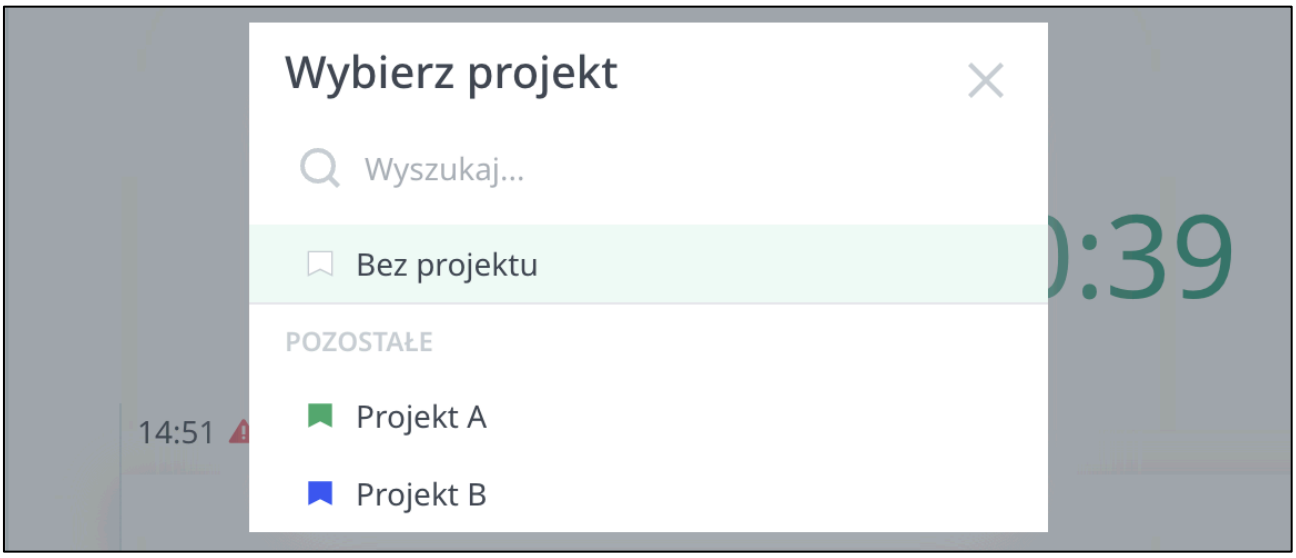
STATUS AFTER RETEST (29.04.2026)

The vulnerability has been partially fixed.

Example #1

The vulnerability has not been fixed.

View from the user interface (the lack of project number 3):



Direct request resulting in access to the project number 3:

```
POST /webapi/clockin/workloging/from-beginning HTTP/2
Host: [DOMAIN]
Cookie: […]
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
Content-Type: application/json
Content-Length: 15
Origin: https://[DOMAIN_2]
Referer: https://[DOMAIN_2]/
Priority: u=0
Te: trailers

{"projectId":3}
```

Server response:

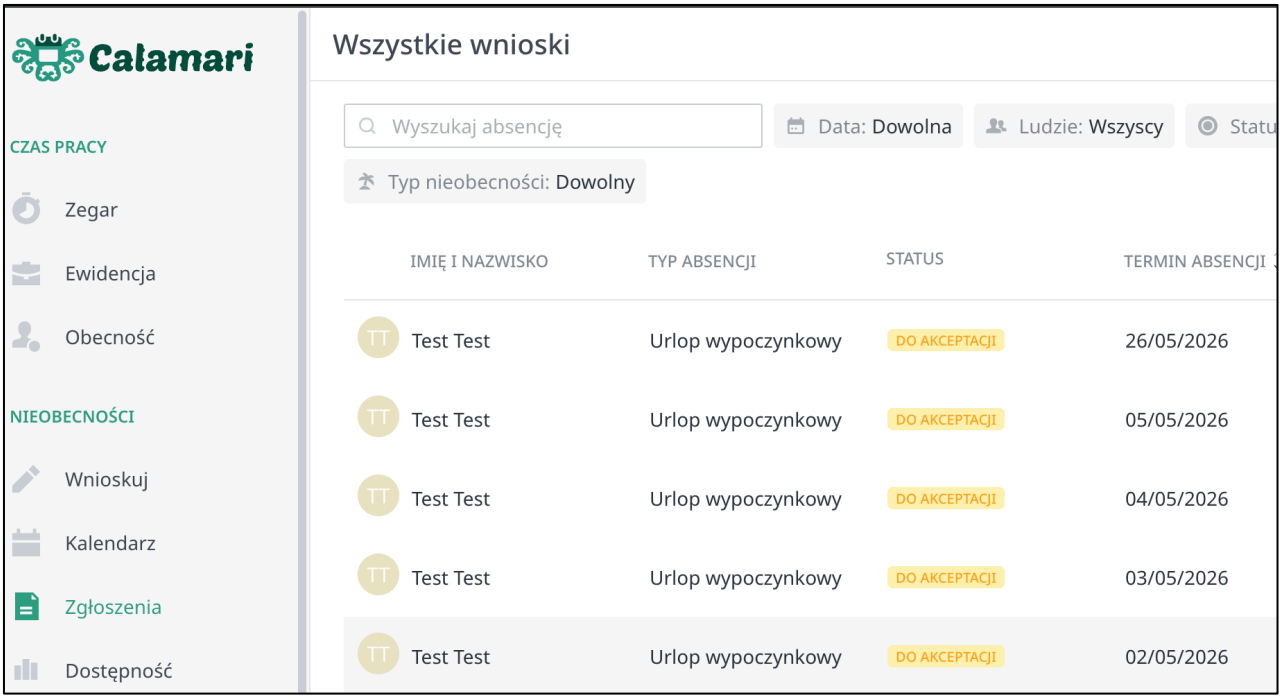
```
HTTP/2 200 OK
Date: Tue, 28 Apr 2026 12:57:21 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 112
Set-Cookie: […]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN_2]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server:
Strict-Transport-Security: max-age=31536000

{"projectId":3,"projectName":"test","color":"#b63eff","startTime":"14:56:50.063","endTime":null,"
previous":null}
```

Example #2

The vulnerability has not been fixed.

View from the user interface (“requests/my”):

View after direct navigation (“requests/all”):



Example #3

The vulnerability has been fixed.

Example #4

The vulnerability has been fixed.

Example #5

The vulnerability has been fixed.

SUMMARY

The tested application does not implement proper authorization of access to data; thus any application user may access data of other users with read privileges.

By exploiting this vulnerability, it was possible to access:

- The details of another project,
- Lists of all requests from the employee level,
- The details of abnormalities and working time rounding
- Reports.

More details:

- https://owasp.org/www-community/Broken_Access_Control
- https://cwe.mitre.org/data/definitions/284.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html

PREREQUISITES FOR THE ATTACK

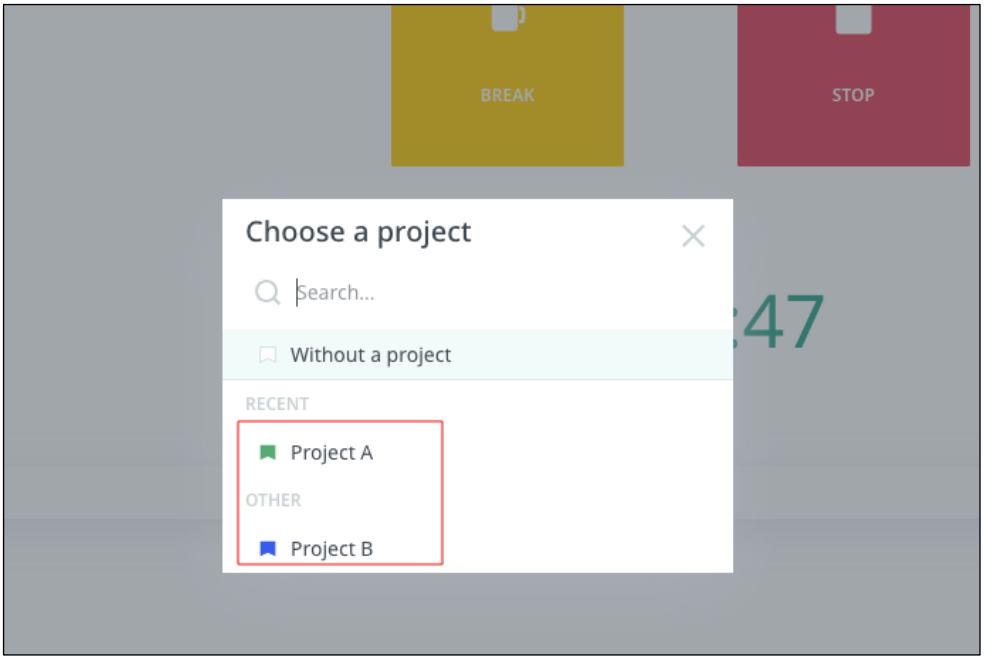
Account in the application (employee role).

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – Projects

In order to gain an access to another user's data, the following steps have to be performed:

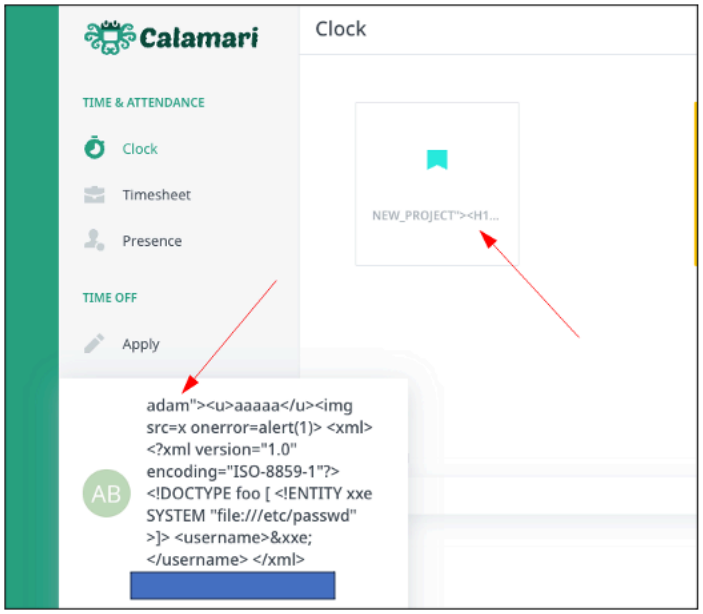
1. Log into the application using employee account.
2. Go to "Clock" and click "Start".
3. Note that only two projects are available:



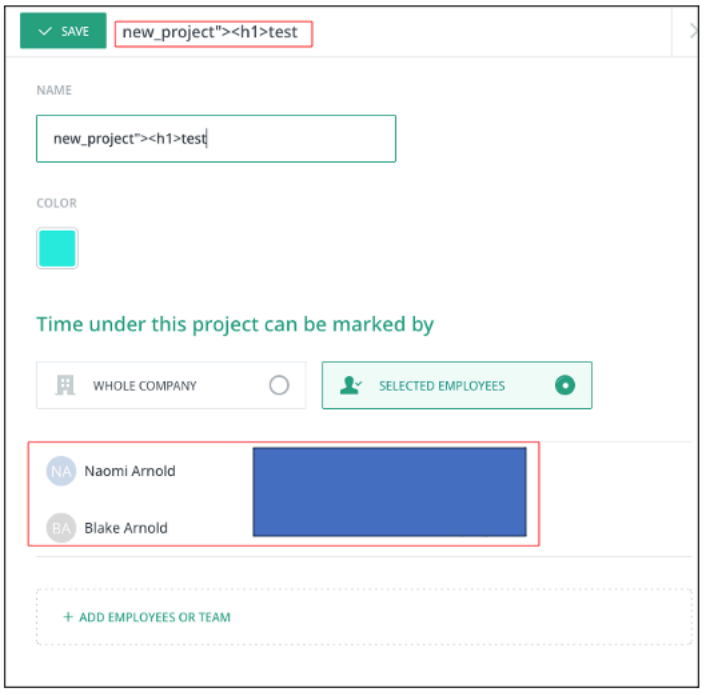
4. Click "Project A". The following HTTP request is sent:

```
POST /webapi/clockin/workloging/from-beginning HTTP/2
Host: [DOMAIN]
Cookie: […]
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
Content-Type: application/json
Content-Length: 15
Origin: https://[DOMAIN_2]
Referer: https://[DOMAIN_2]/
Priority: u=0
Te: trailers

{"projectId":1}
```

5. Change the project ID from 1 to 3 (highlighted in yellow in step 4) and submit the request.
6. Even though the project was not on the list, it was possible to select it by modifying the request:



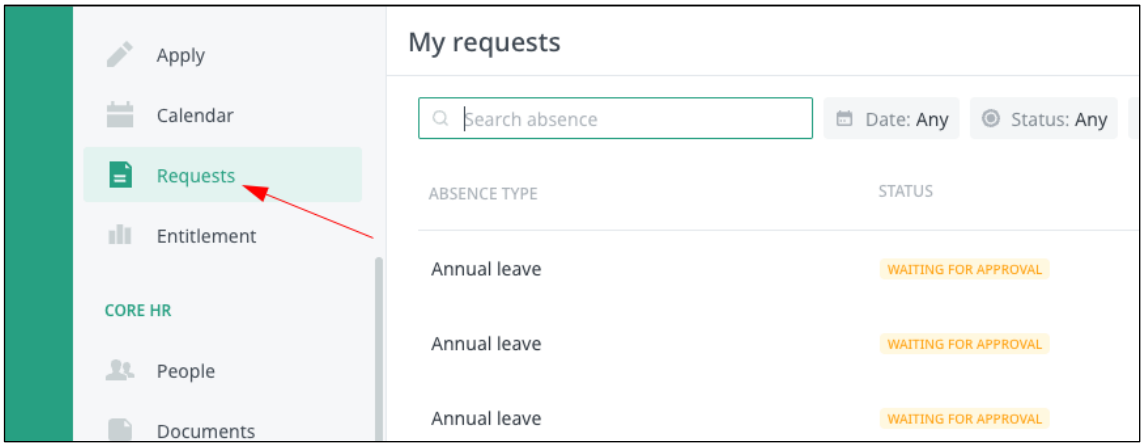
7. The screenshot below shows the view from the project configuration panel available to the administrator and confirms that the user is not on the list of people assigned to the project:



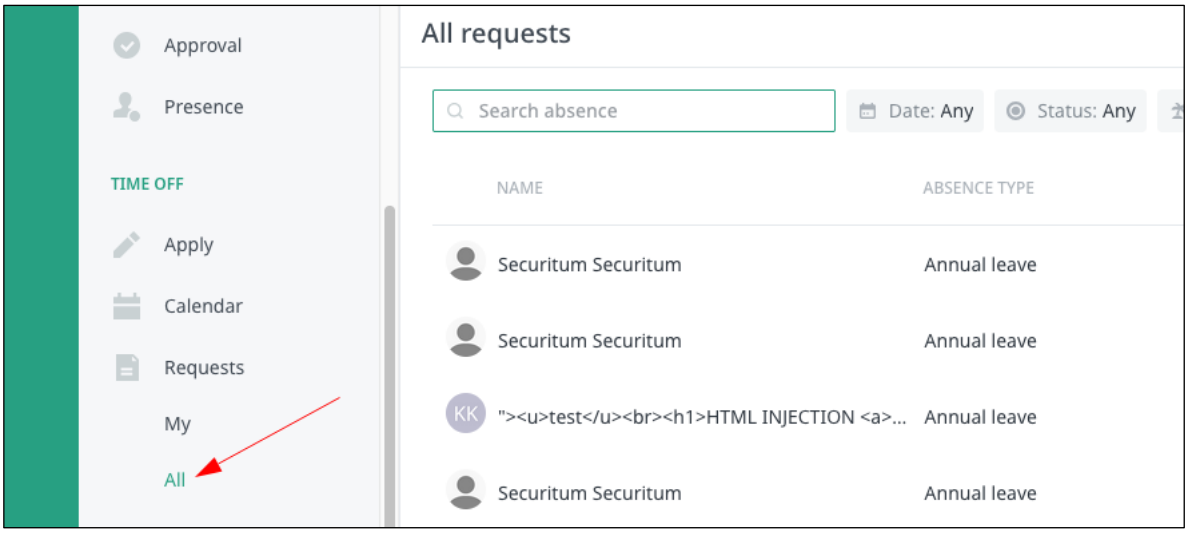
Example #2 – All Requests tab

In order to confirm the vulnerability, the following steps have to be taken:

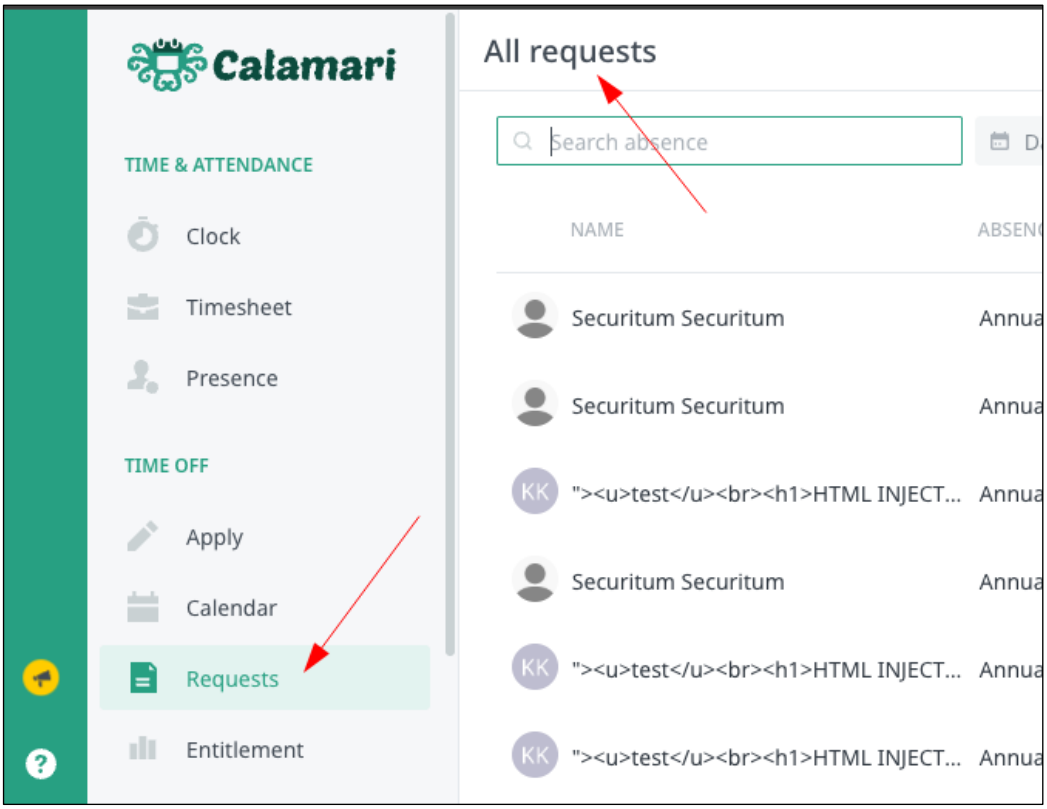
1. Log in to the application (as an employee).
2. Note that user can only see his own requests (lack of “All” tab):



3. For comparison, the view from the administrator's perspective is as follows:



4. After navigating to the `/absence/requests/all` path from an employee level, it is possible to display all requests, even though the option is not available from the GUI level:



Example #3 – Abnormalities

In order to confirm the vulnerability, the following steps have to be taken:

1. Log in to the application (as an employee). The user does not have access to the "Configuration" tab.
2. Send the following HTTP request ("Configuration" -> "Abnormalities"):

```
POST /webapi/clockin/alarms/get HTTP/2
Host: [DOMAIN]
Cookie: [employee_cookie]
User-Agent: Mozilla[…]
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
X-Calamari-Webapp-Request: 1
X-Requested-With: XMLHttpRequest
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/absence/main-new.do?iframe=true
Content-Length: 0
Te: trailers
```

3. Server response:

```
HTTP/2 200 OK
Date: Fri, 26 Sep 2025 13:21:31 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 346
[…]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

```
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

[{"id":2,"type":"EARLY_FINISH","enabled":true,"value":10},{"id":5,"type":"EARLY_START","enabled":true,"value":10},{"id":1,"type":"LATE","enabled":true,"value":10},{"id":6,"type":"WORKED_TOO_LONG","enabled":true,"value":10},{"id":3,"type":"WORKED_TOO_SHORT","enabled":true,"value":10},{"id":4,"type":"WORK_IN_DAY_OFF","enabled":true,"value":null}]
```

4. Attempting to open the "Abnormalities" tab from a web browser returns the following result:

Example #4 – Working time rounding

In order to confirm the vulnerability, the following steps have to be taken:

1. Log in to the application (as an employee). The user does not have access to the "Configuration" tab.
2. Send the following HTTP request ("Configuration" -> "Working time rounding"):

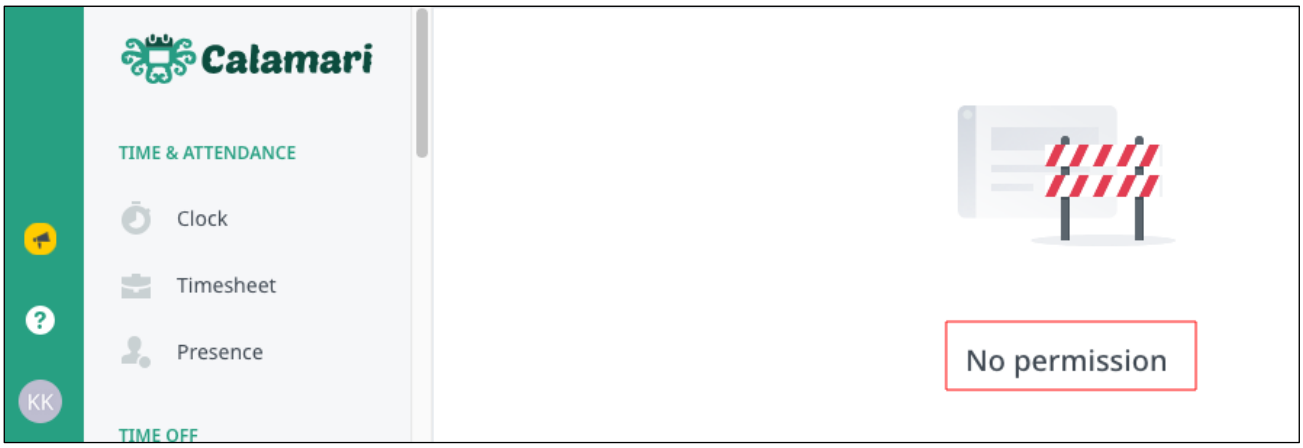
```
GET /webapi/shift-rounding/configuration HTTP/2
Host: [DOMAIN]
Cookie: [employee_cookie]
User-Agent: Mozilla[…]
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
X-Calamari-Webapp-Request: 1
X-Requested-With: XMLHttpRequest
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/absence/main-new.do?iframe=true
Content-Length: 0
Te: trailers
```

3. Server response:

```
HTTP/2 200 OK
Date: Fri, 26 Sep 2025 13:26:46 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 212
[…]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

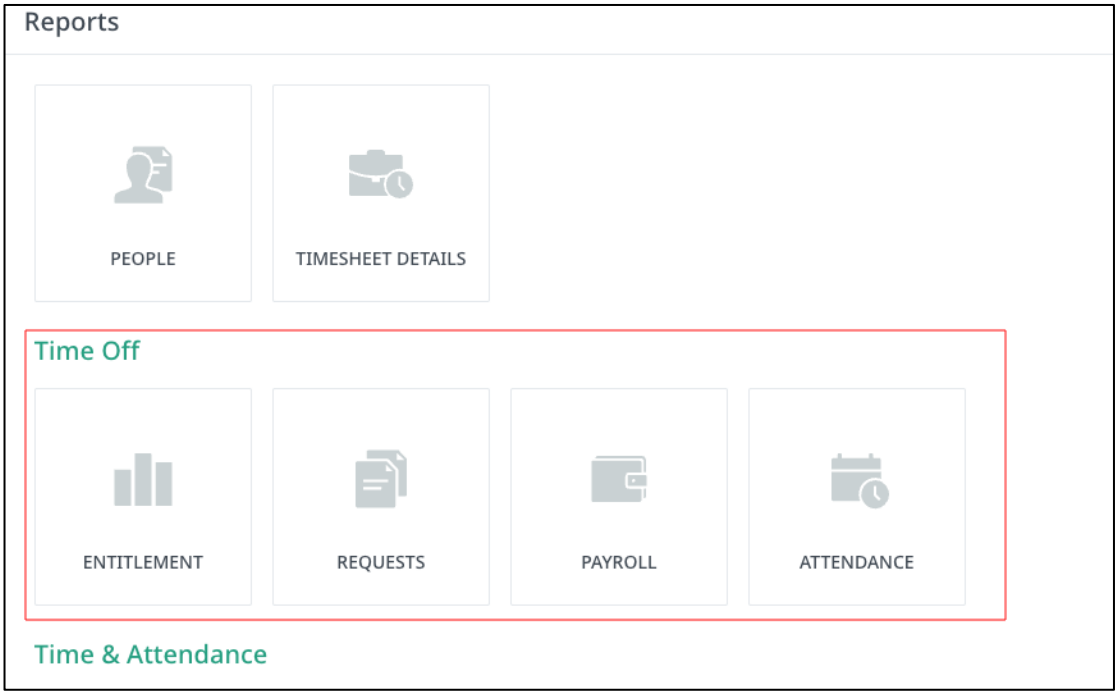
```
Access-Control-Allow-Origin: https://[DOMAIN_2]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

{"wholeOrganization":true,"teams":[],"people":[],"roundingToDayReducingLeavePool":null,"roundingToInterval":{"clockIn":15,"clockInRoundingDirection":"NEAREST","clockOut":15,"clockOutRoundingDirection":"NEAREST"}}
```

4. Attempt to open the " Working time rounding" tab from a web browser returns the following result:

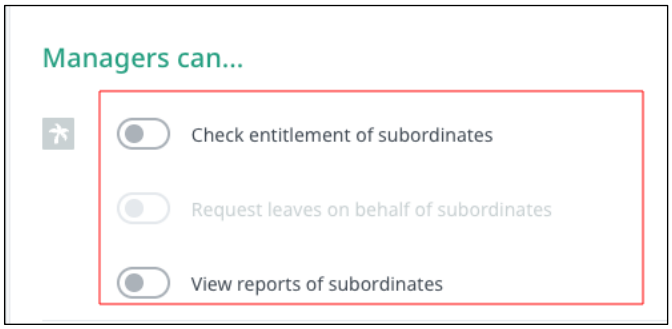


Example #5 – Reports

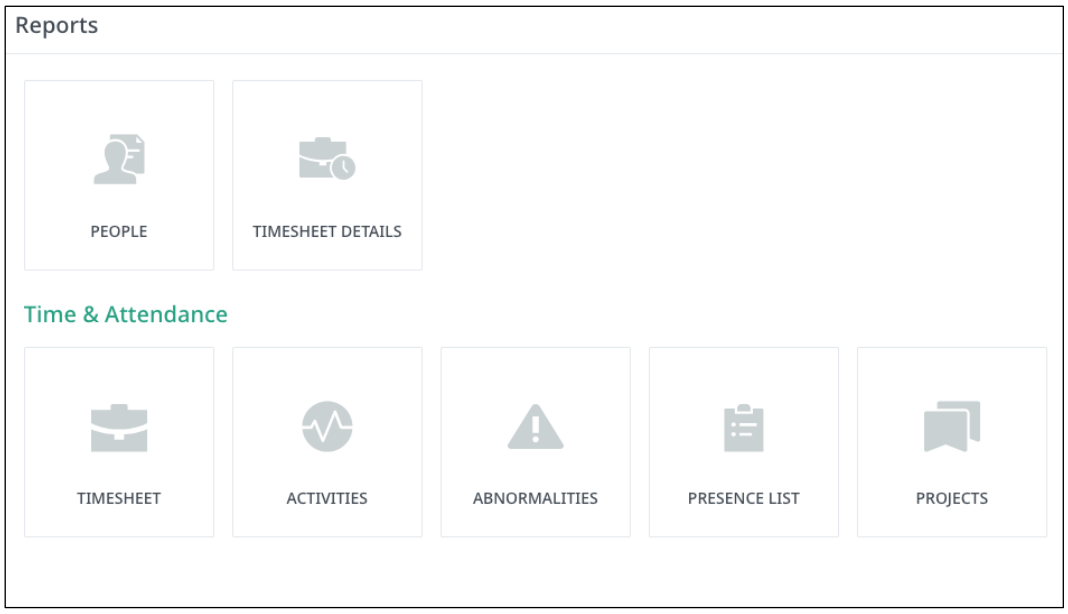
Report's view from the administrator level:



Manager permissions view from the administrator level:

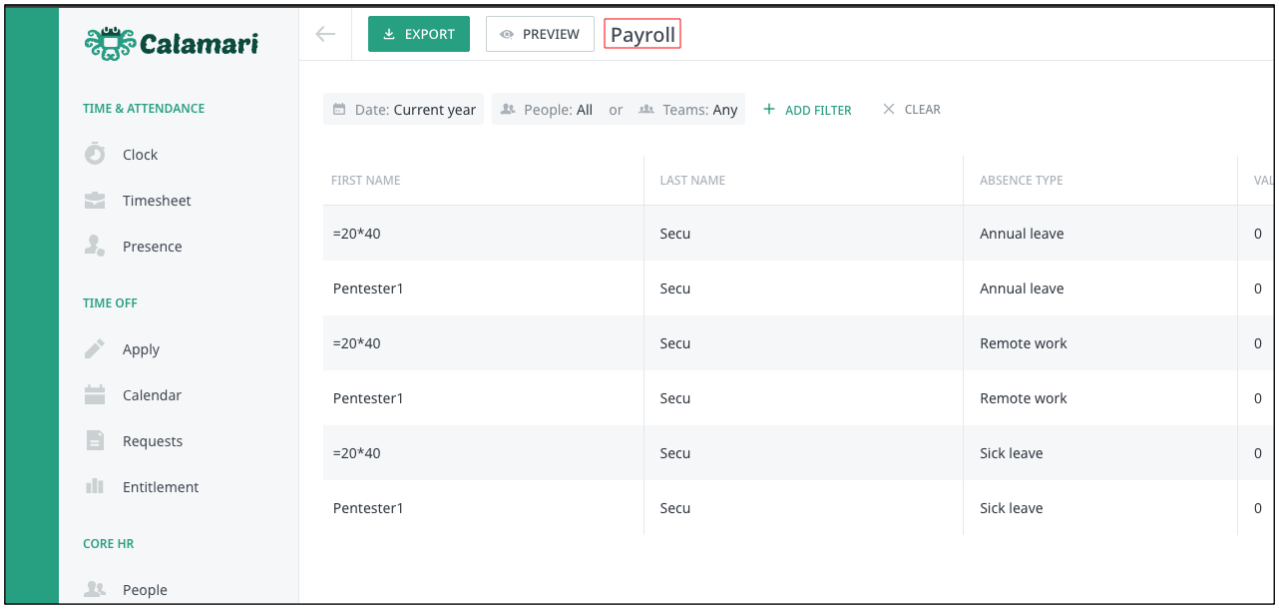


Report's view from the team manager level:



By directly opening the link below from the team manager level, it is possible to navigate to the “Payroll” reports:

```
https://[DOMAIN_2]/clockin/reports/new-payroll
```



The scenarios presented are only examples. Verifying authorization throughout the application is recommended.

LOCATION

Example #1

- https://[DOMAIN]/webapi/clockin/workloging/from-beginning

Example #2

- https://[DOMAIN_2]/absence/requests/all

Example #3

- https://[DOMAIN]/webapi/clockin/alarms/get

Example #4

- https://[DOMAIN]/webapi/shift-rounding/configuration

Example #5

- https://[DOMAIN_2]/clockin/reports/new-payroll

RECOMMENDATION

It is recommended to implement or improve the mechanism responsible for verification of access to data. A user should be able to access only the resources that he or she owns.

It is advisable to use one central authorization module and implement the application so that all operations performed in the application pass through it.

More information:

- https://wiki.owasp.org/index.php/Category:Access_Control
- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Testing_Automation.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V8.2.2 – Verify that the application ensures that data-specific access is restricted to consumers with explicit permissions to specific data items to mitigate insecure direct object reference (IDOR) and broken object level authorization (BOLA).

[FIXED][MEDIUM] SECURITUM-2418465-002: Race condition – possibility of exceeding vacation days

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

During the test, a vulnerability was identified that allowed requests for more vacation days than allowed by the administrator. A race condition is a software flaw where the outcome of a program depends on the timing of events in a multi-threaded or multi-process environment. It can lead to unexpected results when multiple operations compete for shared resources. In a security context, race conditions can be exploited to gain unauthorized access or cause unintended behaviour.

PREREQUISITES FOR THE ATTACK

Account in the application (employee).

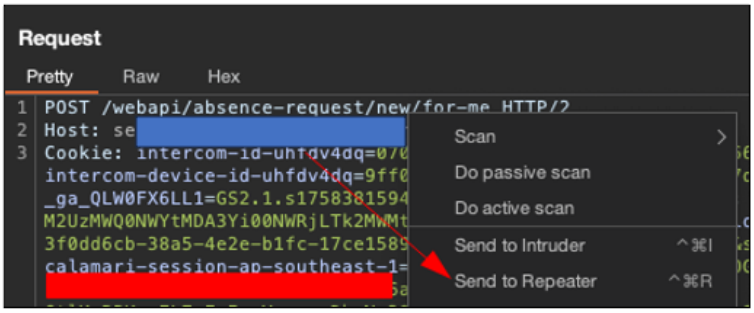
TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to confirm the vulnerability, the following steps have to be taken:

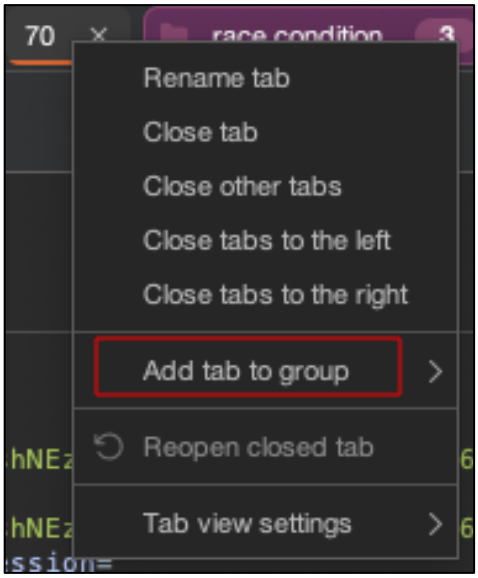
1. Log in to the application as an employee.
2. Go to “Apply”.
3. Select a vacation period and click “OK”. The following request is sent:

```
POST /webapi/absence-request/new/for-me HTTP/2
Host: [DOMAIN]
Cookie: […]
Content-Length: 163
X-Csrf-Token: […]
Accept-Language: en-US,en;q=0.9
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla[…]
Accept: application/json, text/javascript, */*; q=0.01
Content-Type: application/json
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/absence/main-new.do?iframe=true
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"absenceTypeId":"1","absenceStart":"2025-09-24","absenceEnd":"2025-09-
30","reason":"","substitute":"","comment":"","fromPart":null,"toPart":null,"attachments":[]}
```

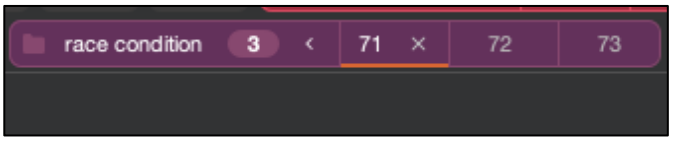
4. Send it to “Repeater” tab:



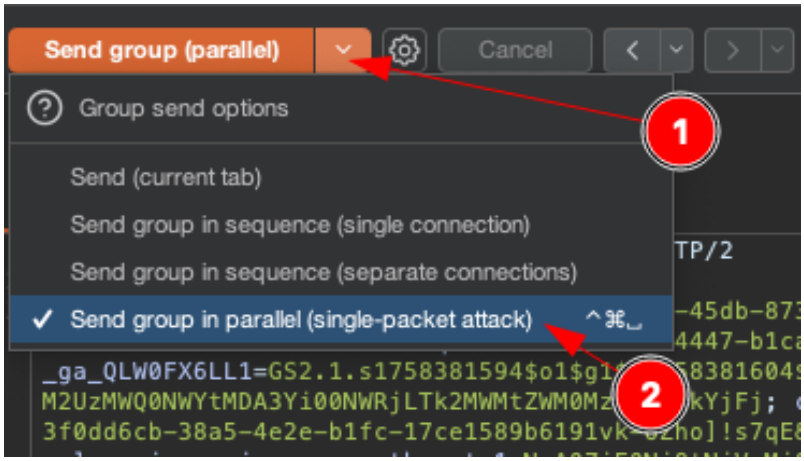
5. Open the “Repeater” tab in Burp Suite.
6. Create tab group:



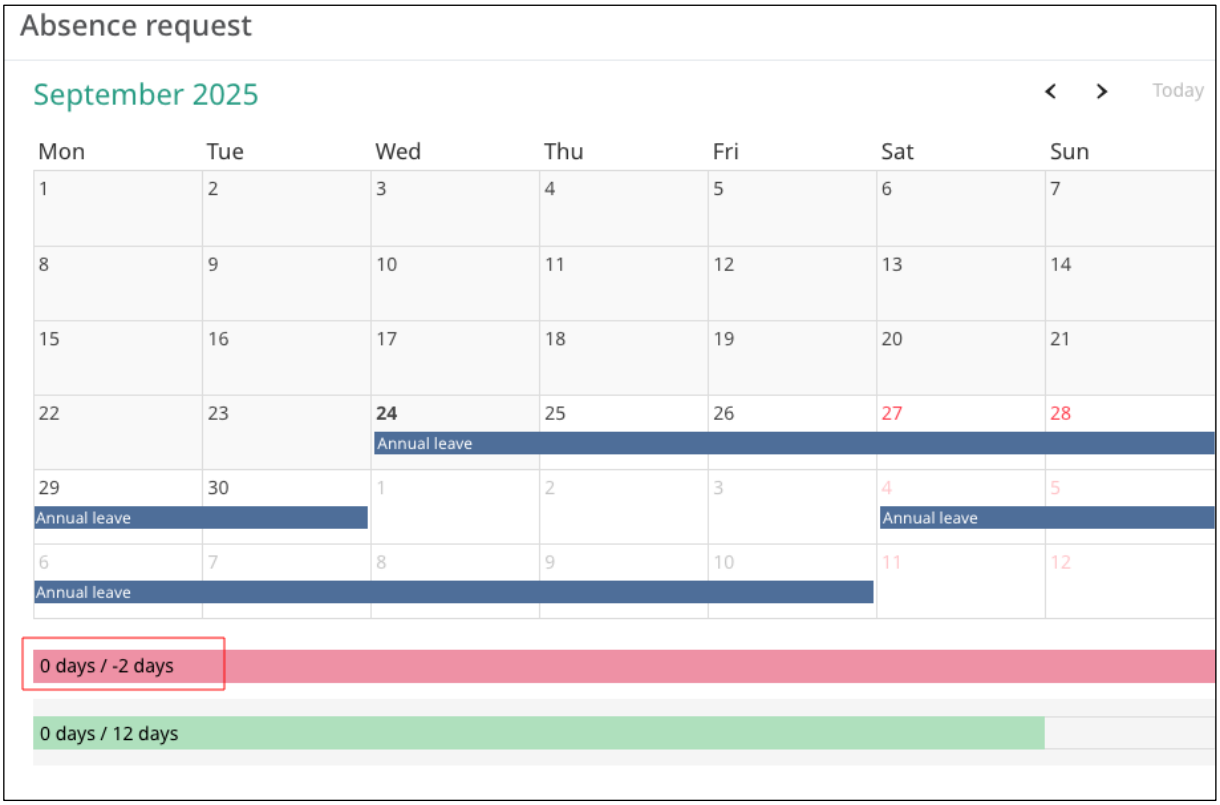
7. In the “Repeater” tab, repeat step 4 several times. Below is an example configuration:



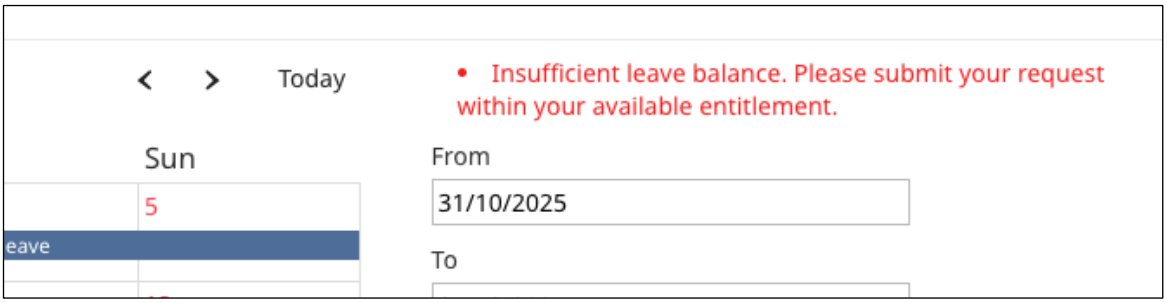
8. Open any request in the new group and click arrow next to “Send” and select “Send group in parallel”:



9. Send group in parallel to perform attack. This will result in three requests being sent in parallel attempting to request an absence. Due to a security flaw, more days will be allocated in absence requests than the employee is entitled to:



10. When trying to send single requests, after using up the available days, the following error is returned:



LOCATION

https://[DOMAIN]/webapi/absence-request/new/for-me

RECOMMENDATION

Employ synchronization mechanisms, such as locks or semaphores, to control access to shared resources. These mechanisms help ensure that only one thread or process can access the resource at a time, preventing race conditions.

Use atomic operations or transactions for operations that involve multiple steps. Atomic operations are executed as a single, indivisible unit, preventing interference from other threads or processes.

Identify critical sections of code where shared resources are accessed and apply synchronization to these specific areas to avoid conflicts.

More information:

- https://portswigger.net/web-security/race-conditions#how-to-prevent-race-condition-vulnerabilities

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V15.4.1 – Verify that shared objects in multi-threaded code (such as caches, files, or in-memory objects accessed by multiple threads) are accessed safely by using thread-safe types and synchronization mechanisms like locks or semaphores to avoid race conditions and data corruption.

[FIXED][MEDIUM] SECURITUM-2418465-004: Denial of Service

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

```
HTTP/2 409 Conflict
Date: Tue, 28 Apr 2026 13:48:16 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 117
Set-Cookie: […]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN_2]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server:
Strict-Transport-Security: max-age=31536000

{"fields":{"dateRange":"Maksymalny zakres dat dla raportu płacowego to 36. miesiące"},"nonfields":[],"warnings":[]}
```

SUMMARY

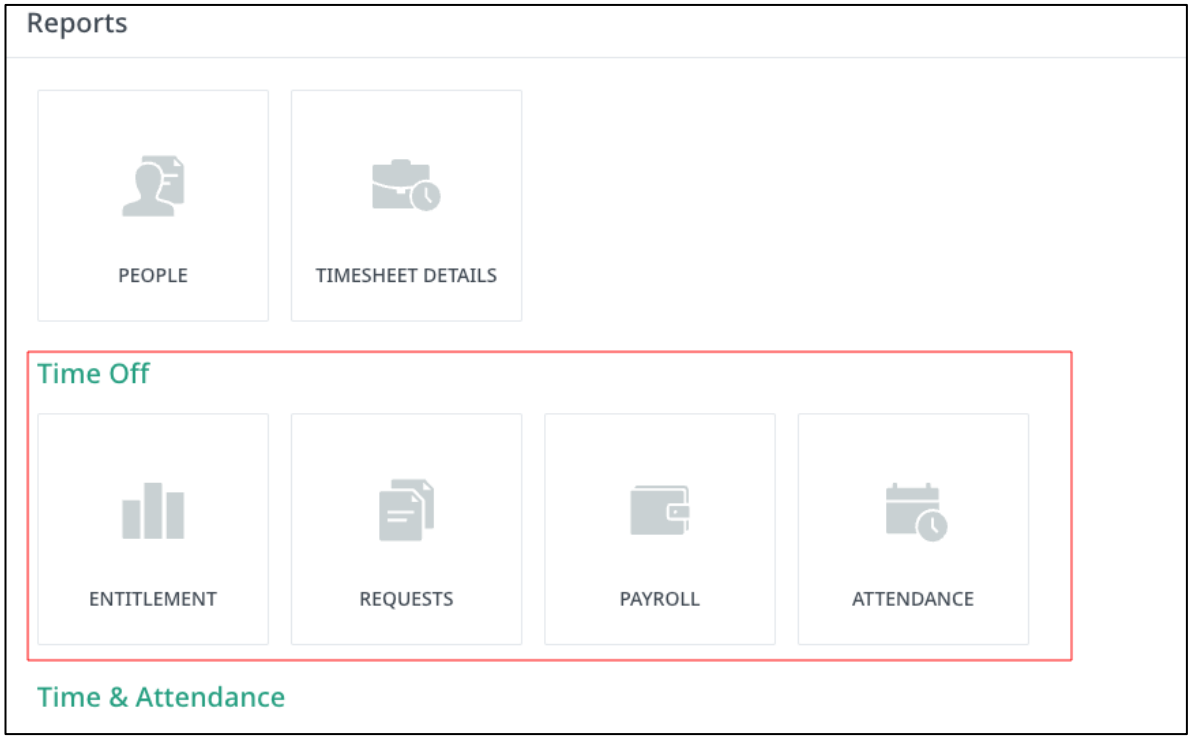
Based on the conducted tests, it was observed that using certain application functionalities significantly extends the response time. Such behaviour can lead to blocking the service.

PREREQUISITES FOR THE ATTACK

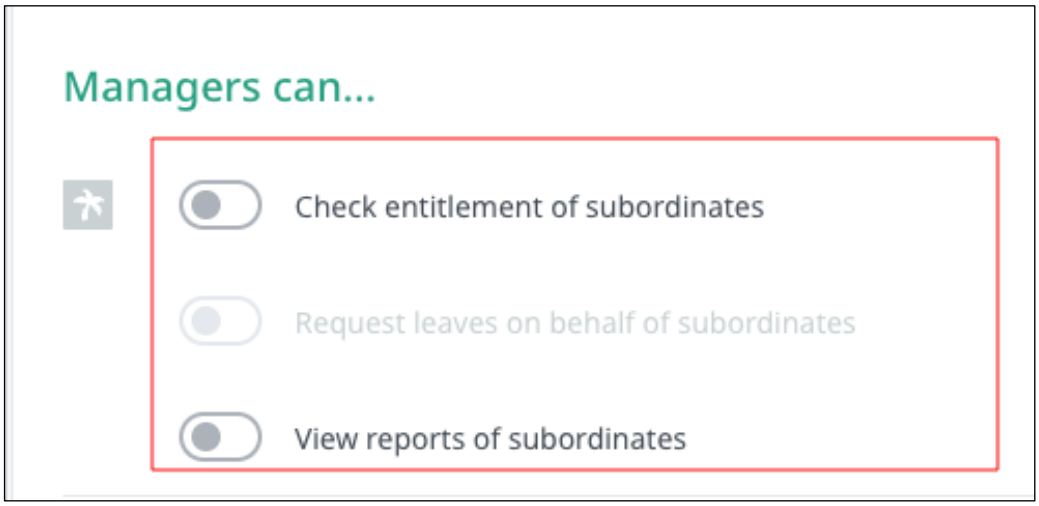
Account in the application (employee).

TECHNICAL DETAILS (PROOF OF CONCEPT)

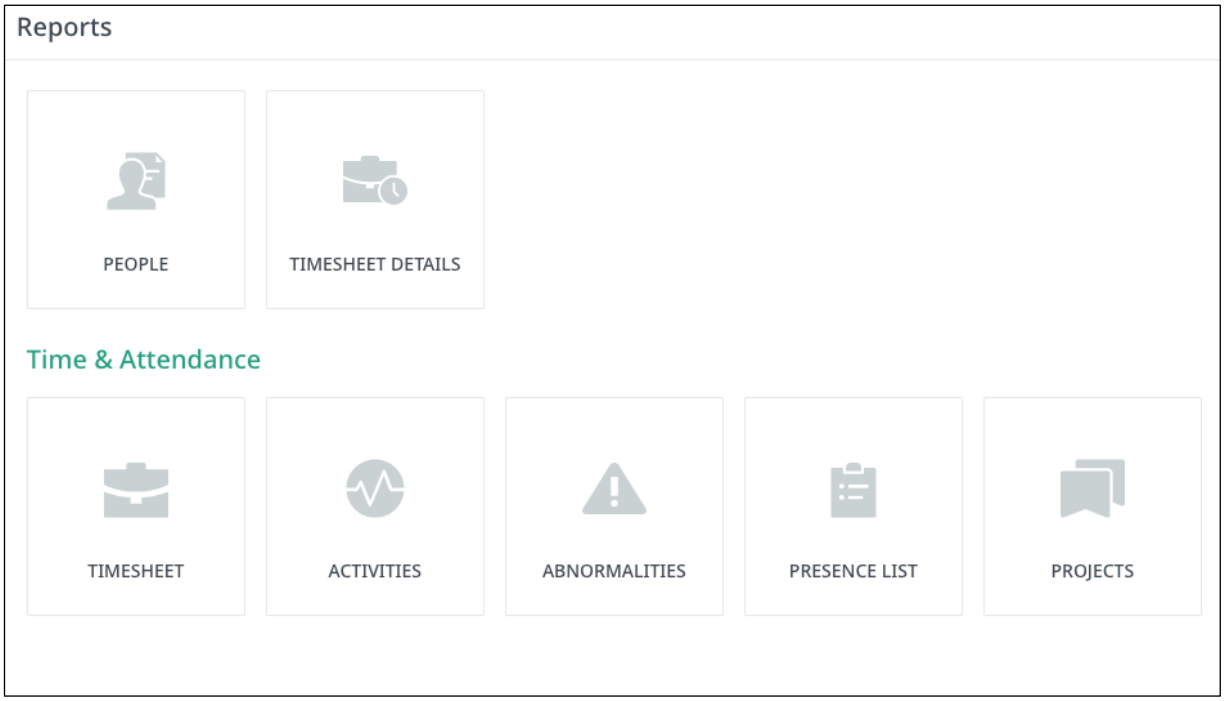
Report's view from the administrator level:



Manager permissions view from the administrator level:

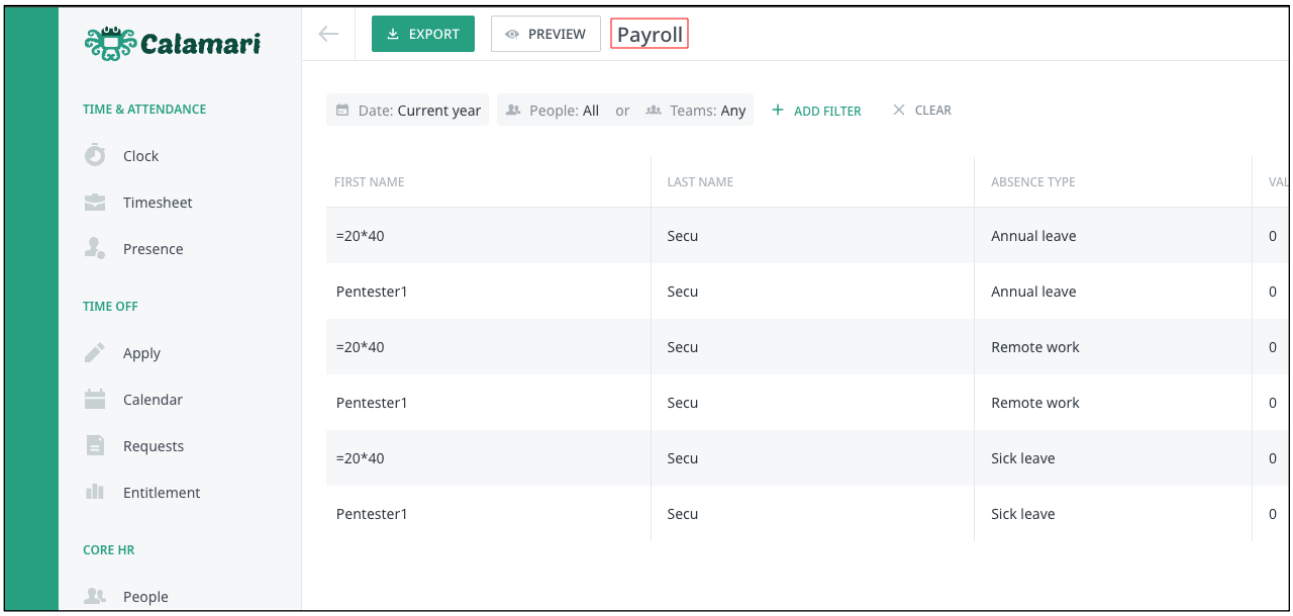


Report's view from the team manager level:



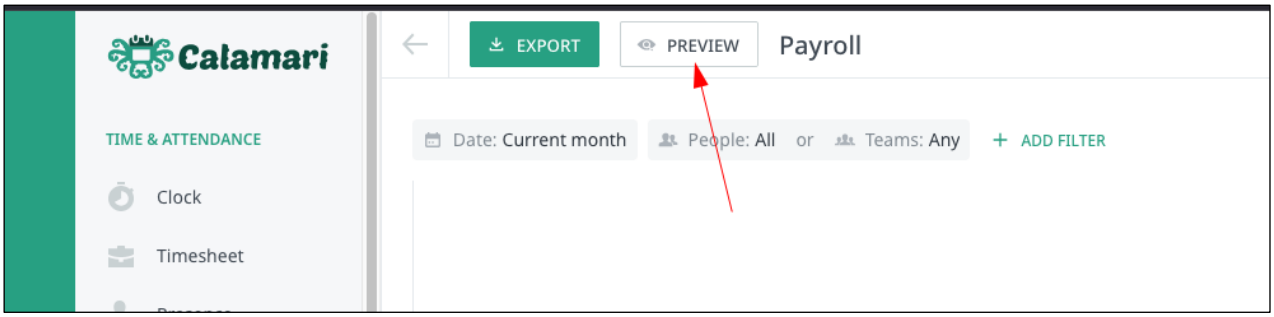
By directly opening the link below from the team manager level, it is possible to go to the “Payroll” reports:

```
https://[DOMAIN_2]/clockin/reports/new-payroll
```



In order to confirm the vulnerability, one needs to:

1. Log in to the application (employee – team manager).
2. Visit the link: https://[DOMAIN_2]/clockin/reports/new-payroll.
3. Click “Preview”:



4. The following request is sent:

```
POST /webapi/leave/report/payroll/get HTTP/2
Host: [DOMAIN]
Cookie: […]
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
Content-Type: application/json
Content-Length: 243
Origin: https://[DOMAIN_2]
Sec-Gpc: 1
Referer: https://[DOMAIN_2]/
Te: trailers

{"dateRange":{"range":"CUSTOM","from":"2025-10-01","to":"2025-10-
31","shift":0},"teams":[],"employees":[],"absenceTypeIds":[],"contractTypes":[],"employeesMode":"
ALL_ACTIVE_AND_MANUALLY_SELECTED_ARCHIVED","groupBy":"ABSENCE_TYPE_MONTHLY_VIEW"}
```

5. In response, the application returns the report data:

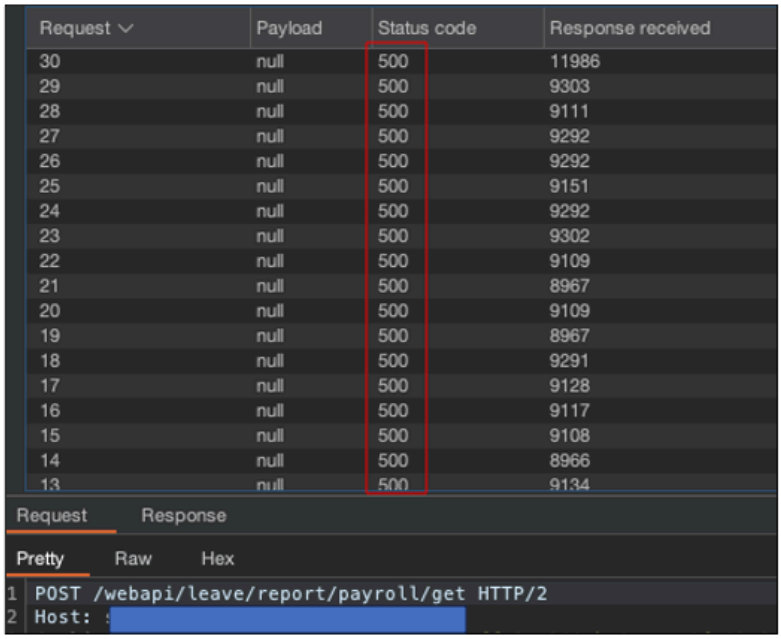
```
HTTP/2 200 OK
Content-Type: application/json;charset=UTF-8
Set-Cookie: […]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN_2]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

{"getAbsenceTypePayrollOuts":[{"employee":{"id":105,"firstName":"\"><u>test</u><br><h1>HTML
INJECTION","lastName":"<a>test1","fullName":"\"><u>test</u><br><h1>HTML                INJECTION
<a>test1","email":"[...]@securitum.pl",[…]
[…]
```

6. Increasing the time range (highlighted in yellow) causes a significant increase in response time:

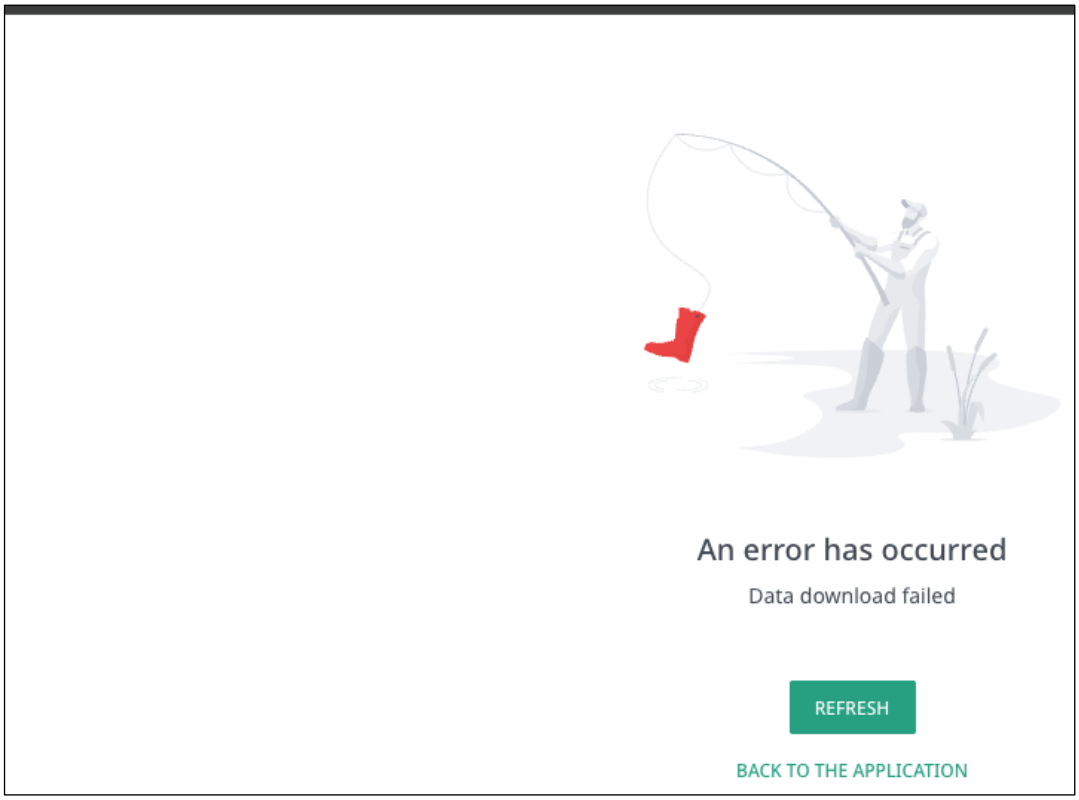
```
POST /webapi/leave/report/payroll/get HTTP/2
```

```
Host: [DOMAIN]
Cookie: […]
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Accept-Language: en
Accept-Encoding: gzip, deflate, br
X-Csrf-Token: […]
Content-Type: application/json
Content-Length: 243
Origin: https://[DOMAIN_2]
Sec-Gpc: 1
Referer: https://[DOMAIN_2]/
Te: trailers

{"dateRange":{"range":"CUSTOM","from":"2015-10-01","to":"2025-10-
31","shift":0},"teams":[],"employees":[],"absenceTypeIds":[],"contractTypes":[],"employeesMode":"
ALL_ACTIVE_AND_MANUALLY_SELECTED_ARCHIVED","groupBy":"ABSENCE_TYPE_MONTHLY_VIEW"}
```

7. Submitting 30 requests simultaneously with a date range from 06-10-1015 to the present resulted in access being blocked:



8. View from the web browser:



The user used to generate the report during the test did not have the appropriate permissions. Access was gained by exploiting a vulnerability related to resource access authorization (SECURITUM-2418465-001).

LOCATION

Reports.

RECOMMENDATION

It is recommended to verify the cause of presented behaviour and eliminate it. It should be ensured that this behaviour does not pose any other security threat.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V2.4.1 – Verify that anti-automation controls are in place to protect against excessive calls to application functions that could lead to data exfiltration, garbage-data creation, quota exhaustion, rate-limit breaches, denial-of-service, or overuse of costly resources.

[FIXED][LOW] SECURITUM-2418465-006: Support for outdated TLS ciphers

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

The tested application supports weak TLS ciphers, which are used to set up a secure communication channel. This could pose a risk of compromising or modifying sensitive user data if an attacker eavesdrops network traffic (Man in the Middle attack, MITM).

More information:

- https://cwe.mitre.org/data/definitions/757.html
- https://cwe.mitre.org/data/definitions/326.html

PREREQUISITES FOR THE ATTACK

Performing a Man in the Middle attack.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The server on which the tested application is running supports the following TLS ciphers:

```
TLSv1.2
        ECDHE-RSA-AES256-SHA384
        ECDHE-RSA-AES128-SHA256
```

LOCATION

Server configuration ([DOMAIN_2], [DOMAIN]).

RECOMMENDATION

It is recommended to disable support for the TLS ciphers mentioned above.

The current recommended algorithm configuration can be found at:

- https://ssl-config.mozilla.org/

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V12.1.2 – Verify that only recommended cipher suites are enabled, with the strongest cipher suites set as preferred. L3 applications must only support cipher suites which provide forward secrecy.

[FIXED][LOW] SECURITUM-2418465-007: Cookies without SameSite attribute set

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed. Indicated cookie contains the “Lax” attribute now.

SUMMARY

During the tests it was observed that the application does not set the “SameSite” security attribute for important cookies. Setting the “SameSite” attribute could prevent browser from sending a cookie between pages with different origins. Implementing such attribute could be helpful among others in preventing CSRF attacks.

Below there is a list of cookies for which the “SameSite” attribute was not identified:

- calamari.cloud.session.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite
- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#samesite-attribute
- https://wiki.owasp.org/index.php/Testing_for_cookies_attributes_(OTG-SESS-002)
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The screenshot below shows the current cookie configuration:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_csrf_token	af4de45e-dfaf-42...	.c	/	Thu, 23 Oct 2025 ...	47	false	true	None	Wed, 24 Sep 2025 06:41:51 GMT
AWSALBCORS	YP7Lu3JFI0TVqDa...	s	/	Wed, 01 Oct 2025...	134	false	true	None	Wed, 24 Sep 2025 06:43:24 GMT
AWSALB	YP7Lu3JFI0TVqDa...	s	/	Wed, 01 Oct 2025...	130	false	false		Wed, 24 Sep 2025 06:43:24 GMT
calamari-saml-au...	eadb211a-6a53-4...	.c	/	Mon, 22 Sep 2025...	65	true	true		Mon, 22 Sep 2025 12:44:02 GMT
calamari-session-...	MDE0NmYyMmMt...	.c	/	Mon, 20 Oct 2025...	79	true	true		Wed, 24 Sep 2025 06:41:51 GMT
calamari.cloud.se...	e5f9b5d0-f0fd-45...	.c	/	Wed, 22 Oct 2025...	133	true	true	None	Wed, 24 Sep 2025 06:41:51 GMT
intercom-device-i...	f2631bee-510a-4f...	.c	/	Wed, 17 Jun 2026 ...	63	false	false	Lax	Wed, 24 Sep 2025 06:41:51 GMT
intercom-id-uhfd...	da485d44-3a28-4...	.c	/	Wed, 17 Jun 2026 ...	56	false	false	Lax	Wed, 24 Sep 2025 06:41:51 GMT
intercom-session...		.c	/	Sat, 27 Sep 2025 ...	25	false	false	Lax	Wed, 24 Sep 2025 06:41:51 GMT

LOCATION

Cookie management.

RECOMMENDATION

It is recommended that the application sets the "SameSite" attribute for important cookies with one of the following values:

- `Strict` – the browser will not send the cookie with cross-site requests,

- `Lax` – the browser will send the cookie with cross-site requests if and only if they are sent using safe HTTP method (GET, HEAD, OPTIONS, TRACE) and they are top-level navigations (i.e. the address bar will show the change of domain); in other cases of cross-domain requests, the cookie will not be sent.

E.g.:

```
Set-Cookie: foo=bar; SameSite=Strict
```

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.3.2 – Verify that each cookie's "SameSite" attribute value is set according to the purpose of the cookie, to limit exposure to user interface redress attacks and browser-based request forgery attacks, commonly known as cross-site request forgery (CSRF).

[PARTIALLY FIXED][LOW] SECURITUM-2418465-008: Outdated software

STATUS AFTER RETEST (03.06.2026)

The vulnerability has been partially fixed. As indicated by the Client, it is currently not possible to update the library to the latest version due to environment constraints.

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been partially fixed.

Example #1

The vulnerability has been fixed. The Client indicated that the moment.js version was updated to version 2.30.1.

Example #2

Due to changes in the environment, the OpenPDF library is outdated and requires updating (SECURITUM-2418465-010).

SUMMARY

It was observed that some software components are not updated to the newest versions, and it can be found that they contain publicly known vulnerabilities.

During the tests it was not possible to prepare a working Proof of Concept using the described vulnerability, however the mere fact of using software with publicly known vulnerabilities exhausts the necessity to include such information in the report.

More information:

- https://www.cve.org/CVERecord?id=CVE-2022-24785
- https://www.cve.org/CVERecord?id=CVE-2024-29857
- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

PREREQUISITES FOR THE ATTACK

Depends on the vulnerability.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is a table with the version of the current software along with the hosts which it has been identified on:

Example location / URL	Software and version
https://[DOMAIN]/ui-xp/old-part.3d76d6b4798c1f4e0f5e8c473185c2b2.js	Moment.js 2.17.1
PDF files (metadata)	iText 5.5.5

LOCATION

Indicated in the technical details section.

RECOMMENDATION

It is recommended to update the software to the latest, stable version.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V15.2.1 – Verify that the application only contains components which have not breached the documented update and remediation time frames.

[FIXED][LOW] SECURITUM-2418465-009: Modifying error messages

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

During the test, it was observed that users can create their own error messages. This behaviour can be exploited, among other things, to conduct phishing attacks.

PREREQUISITES FOR THE ATTACK

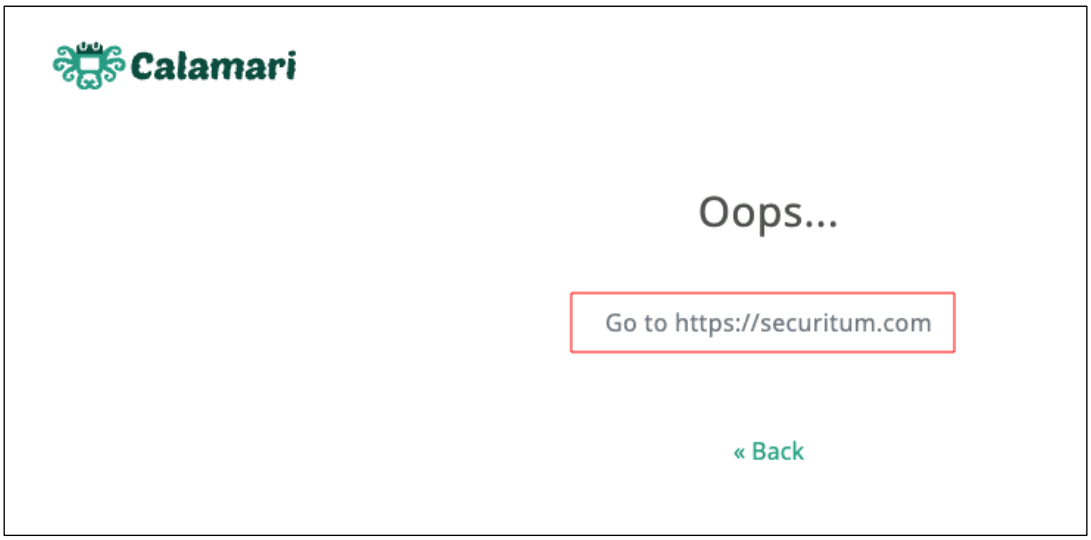
None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below is the URL address that displays an error message when opened:

```
https://[DOMAIN]o/error-callback?error=Go%20to%20https://securitum.com
```

View from a web browser:



LOCATION

https://[DOMAIN]o/error-callback?error=

RECOMMENDATION

It is recommended that the user does not have any influence on the content of the error message.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V16.5.1 – Verify that a generic message is returned to the consumer when an unexpected or security-sensitive error occurs, ensuring no exposure of sensitive internal system data such as stack traces, queries, secret keys, and tokens.

[FIXED][LOW] SECURITUM-2418465-010: Redundant information disclosure about the application environment

STATUS AFTER RETEST (03.06.2026)

The vulnerability has been fixed.

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been partially fixed. While it is no possible to obtain information from Example #2 anymore, it is still possible to obtain PDF metadata i.e. “OpenPDF 2.0.3”.

SUMMARY

During the audit, it was observed that the tested application returns redundant information about the technologies in use. This behaviour can help an attacker to better profile the application environment, which then can be used to carry out further attacks.

PREREQUISITES FOR THE ATTACK

Access to the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – PDF metadata

Below is the metadata for a sample PDF file (Requests -> All -> PDF):

```
ExifTool Version Number         : 12.76
File Name                       : adam___u_aaaaa__u__img src=x onerror=alert(1)__brak_2025-10-
01_68.pdf
Directory                       : .
File Size                       : 49 kB
File Modification Date/Time     : 2025:09:22 07:42:21+02:00
File Access Date/Time           : 2025:09:22 07:43:09+02:00
File Inode Change Date/Time     : 2025:09:22 07:43:09+02:00
File Permissions                : -rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Modify Date                     : 2025:09:22 07:42:21+02:00
Create Date                     : 2025:09:22 07:42:21+02:00
Producer                        : iText® 5.5.5 ©2000-2014 iText Group NV [REDACTED]
Page Count                      : 1
```

Example #2 – Application’s response

Request:

```
OPTIONS /a'a%5c'b%22c%3e%3f%3e%25%7d%7d%25%25%3ec%3c[[%3f$%7b%7b%25%7d%7dcake%5c/personal-
property/save HTTP/2
Host: [DOMAIN]
User-Agent: Mozilla[…]
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type,x-csrf-token
Referer: https://[DOMAIN_2]/
Origin: https://[DOMAIN_2]
Priority: u=4
```

Response:

```
HTTP/2 400 Bad Request
Date: Wed, 24 Sep 2025 14:24:19 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 796
[…]
Content-Language: en

[…]> The server cannot or will not process the request due to something that is perceived to be a
client error (e.g., malformed request syntax, invalid request message framing, or deceptive
request routing).</p><hr class="line" /><h3>Apache Tomcat/9.0.102</h3></body></html>
```

LOCATION

Indicated in the technical details section.

RECOMMENDATION

It is recommended to remove all unnecessary information about the technologies used.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V13.4.6 – Verify that the application does not expose detailed version information of backend components.

[FIXED][LOW] SECURITUM-2418465-011: HTML injection

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

During the test, it was discovered that HTML tags can be injected into emails by sending an invitation to a new user. Once the email is opened, the inserted code is interpreted by the browser.

PREREQUISITES FOR THE ATTACK

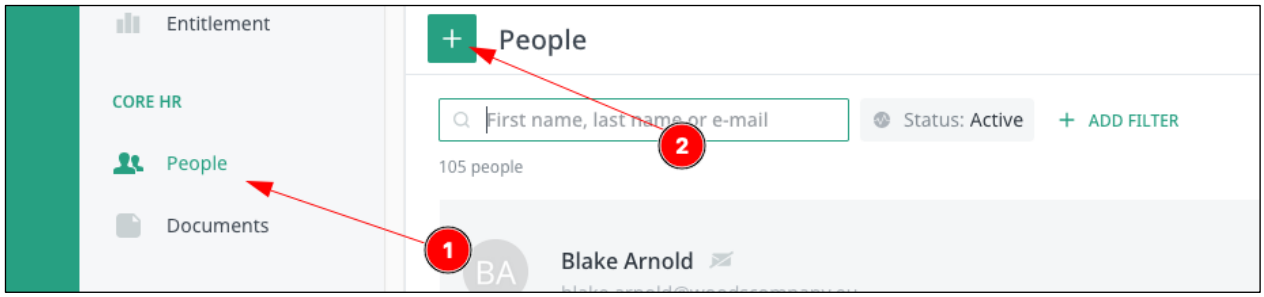
Administrator account.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – Invitations

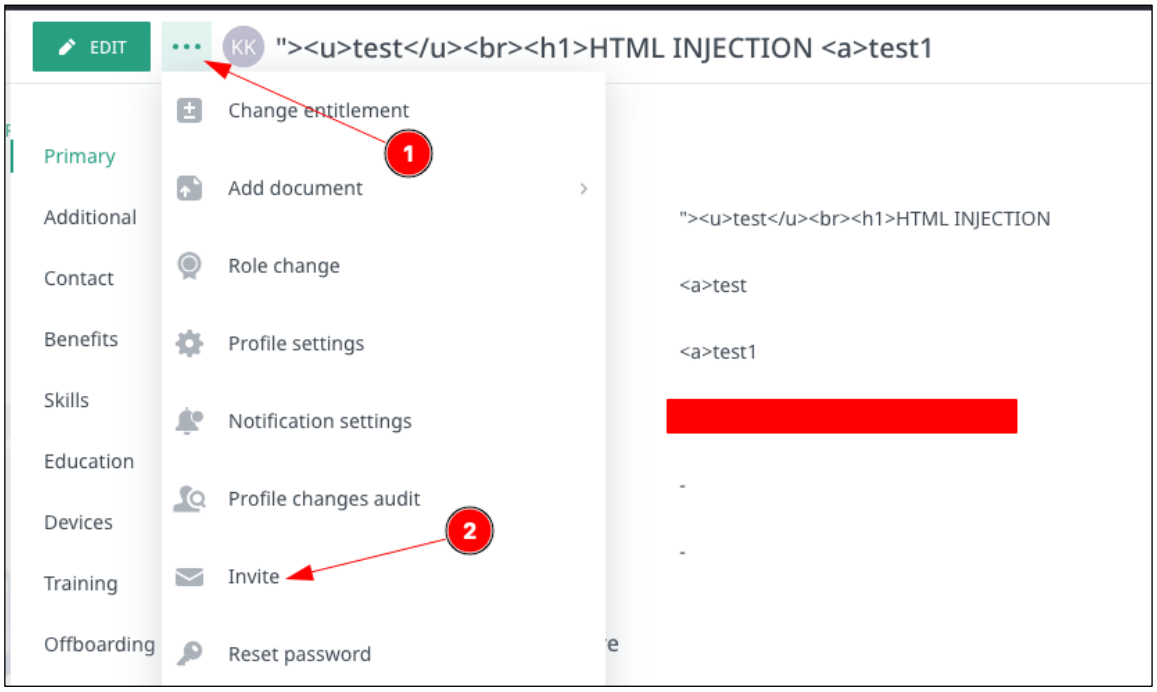
To send an invitation containing a HTML code, one has to follow the steps below:

1. Log in to the application with an administrator account.
2. Go to “People” tab and click “Add”:

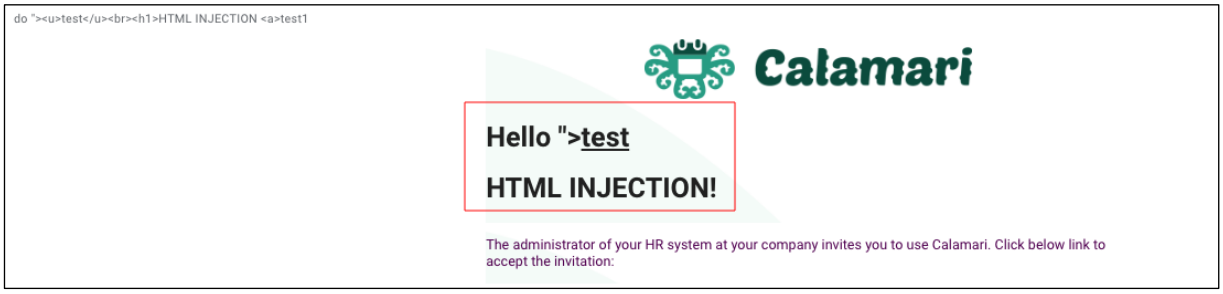


3. Complete the form as below and click “Save”:

4. Click “Invite”:



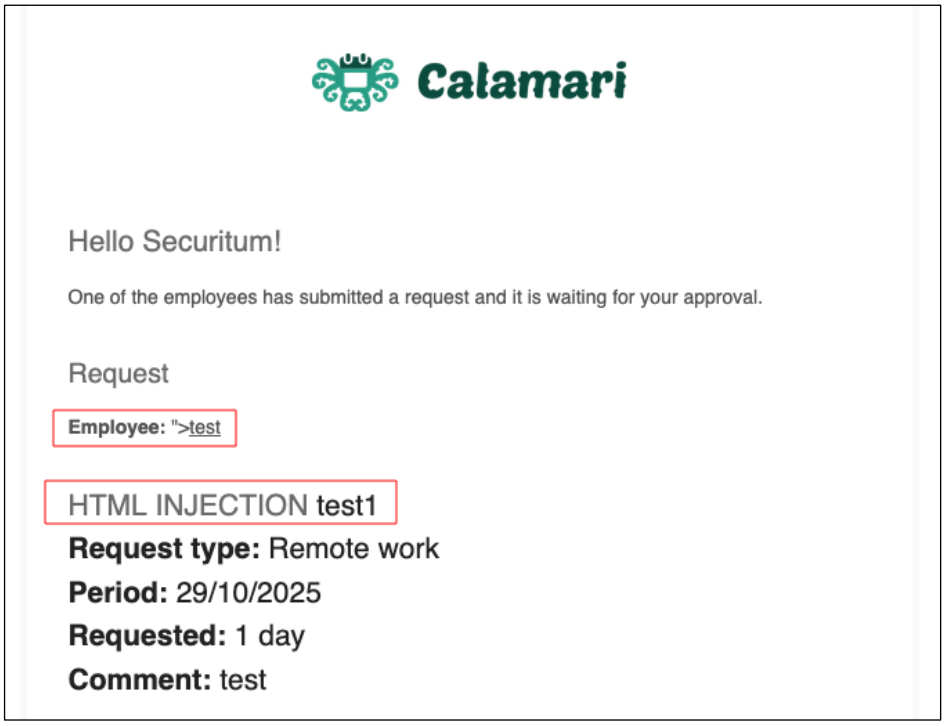
5. When the email is opened, the entered HTML code is interpreted:



Example #2 – Requests

To send an email containing a HTML code, one has to follow the steps below:

1. Log in to the application with an employee account.
2. Go to “Apply” tab and send a new request.
3. Click “OK”.
4. The person responsible for approving the request then receives an email with this information. The content includes interpreted HTML code:



LOCATION

Example #1

- Email invitations

Example #2

- Requests

RECOMMENDATION

It is recommended to validate all data received from the user (to reject the values that are inconsistent with the template/format of a given field – whitelist approach), and then encode it on the output in relation to the context in which it is embedded (in all places of the application, not only those specified in the description).

It is recommended to implement the mechanism of filtering the MIMEtype in a request and make it not possible to inject HTML/JavaScript code into the event title. In case if one would find a way to bypass the DOMPurify library and CSP, which are used to secure the application against XSS attacks, one could potentially exploit it.

For this purpose, it should be verified whether the framework used by the application has built-in functions that implement the described recommendation.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V1.2.1 – Verify that output encoding for an HTTP response, HTML document, or XML document is relevant for the context required, such as encoding the relevant characters for HTML elements, HTML attributes, HTML comments, CSS, or HTTP header fields, to avoid changing the message or document structure.

[FIXED][LOW] SECURITUM-2418465-013: Open Redirect – possibility to redirect a user to a malicious domain

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

The analysis showed that the application does not correctly validate the URL to which a user is being redirected. Using this fact, an attacker, may send the user to a malicious page.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

PREREQUISITES FOR THE ATTACK

Victim must click the link.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The following is an example of a request that includes the URL to which the redirect takes place (disabling integration in the admin panel):

```
GET
/webapi/integration/google/unintegrate?directly_back_to=https://securitum.com/clockin/configurati
on/google HTTP/2
Host: pentest2025.asia.calamari.io
Cookie: […]
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla[…]
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.7
Referer: https://[DOMAIN_2]/
```

In response, the application confirms the acceptance of the address and performs the redirection:

```
HTTP/2 307 Temporary Redirect
Date: Mon, 29 Sep 2025 13:48:02 GMT
Content-Length: 0
Location: https://securitum.com/clockin/configuration/google
[…]
```

LOCATION

- https://pentest2025.[DOMAIN]webapi/integration/google/unintegrate?directly_back_to=[...]
- https://pentest2025.[DOMAIN]webapi/integration/office/unintegrate?directly_back_to=[...]

RECOMMENDATION

It is recommended to verify the destination address to which the redirection takes place, e.g. by creating a list of allowed addresses to which users can be redirected (validation should take place on the server side).

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.7.2 – Verify that the application will only automatically redirect the user to a different hostname or domain (which is not controlled by the application) where the destination appears on an allowlist.

[NOT VERIFIED][LOW] SECURITUM-2418465-014: Basic Authentication

SUMMARY

The tested application uses HTTP Basic Authentication. Several security risks are associated with this authentication method:

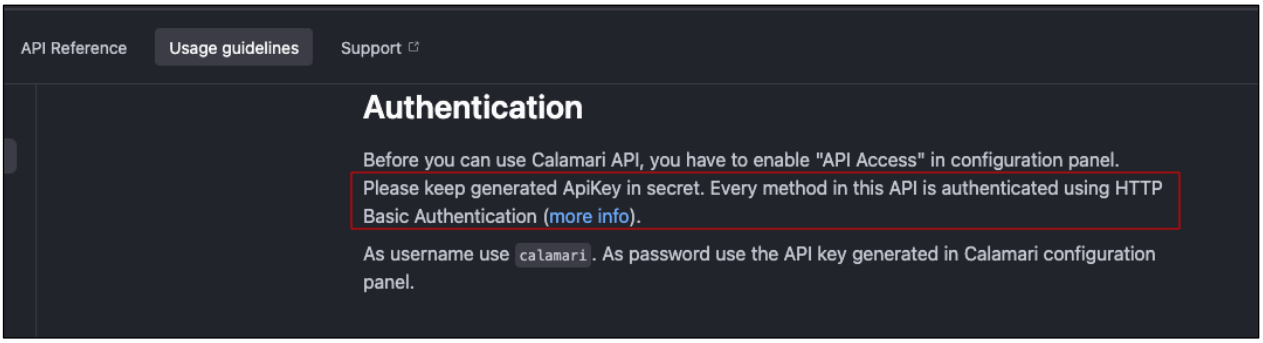
1. Increased Attack Surface due to Repetitive Transmission - The authentication credentials must be transmitted with every single HTTP request. This significantly increases the potential attack window (or "attack surface") because the same sensitive data appears in multiple locations, even when using an encrypted communication channel (HTTPS).
2. Lack of Simple Logout Mechanism - Once credentials are sent to the application via HTTP Basic Authentication, all web browsers store this information until the browser window is fully closed. This means there is no straightforward way for a user to actively "log out" or end their session manually within the application interface.
3. Vulnerability to Local Malware and Credential Theft - In the case of HTTP Basic Authentication, there is no simple way to restrict the browser from saving the password in its local database or cache. This creates a risk of credentials being compromised by malware running on an infected user machine.
4. Risk of Shared Credentials and Lack of Accountability - If the web server hosting the application is not configured to use a central user management system, there is a risk of establishing a single, shared login and password pair among multiple users. This is an unacceptable practice, primarily because it makes it difficult to ascertain who accessed the system at a given time (lack of accountability).
5. Ineffective Brute-Force Protection Leading to Denial of Service (DoS) - A web server using HTTP Basic Authentication is not protected against brute-force attacks by default. Implementing such a solution requires additional configuration, which may still fail when using shared credentials. Automatically blocking an account for a specified time (e.g., using a tool like fail2ban) can lead to a Denial of Service (DoS) for all other users relying on that same shared credential.
6. Missing Password Change Functionality - The application lacks a password change feature. In the event of a user's credentials being compromised or leaked, there is no quick or simple way to change the access data, leaving the account permanently vulnerable until manual intervention.
7. No Session Control or Connection Monitoring - It is possible to connect to the application multiple times simultaneously using the same credentials. The system provides no information about currently active connections or the ability to terminate specific sessions if suspicious activity is detected.

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The screenshot below demonstrates the API documentation:



LOCATION

API authentication.

RECOMMENDATION

It is recommended to discontinue use of HTTP Basic Authentication. Client certificates can be used to further secure access to the panel.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V6.3.1 – Verify that controls to prevent attacks such as credential stuffing and password brute force are implemented according to the application's security documentation.
V7.2.2 – Verify that the application uses either self-contained or reference tokens that are dynamically generated for session management, i.e. not using static API secrets and keys.

[FIXED][LOW] SECURITUM-2418465-015: Accessing the local administrator account without proper permissions

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

During the test, it was discovered that it was possible to gain access to the administrator account using the account specified during Google Workspace integration. This allows for the escalation of privileges for a standard user and the ability to make configuration changes as an administrator.

It is also important that the integration can be configured using a regular Google Workspace user, one who does not have administrator privileges.

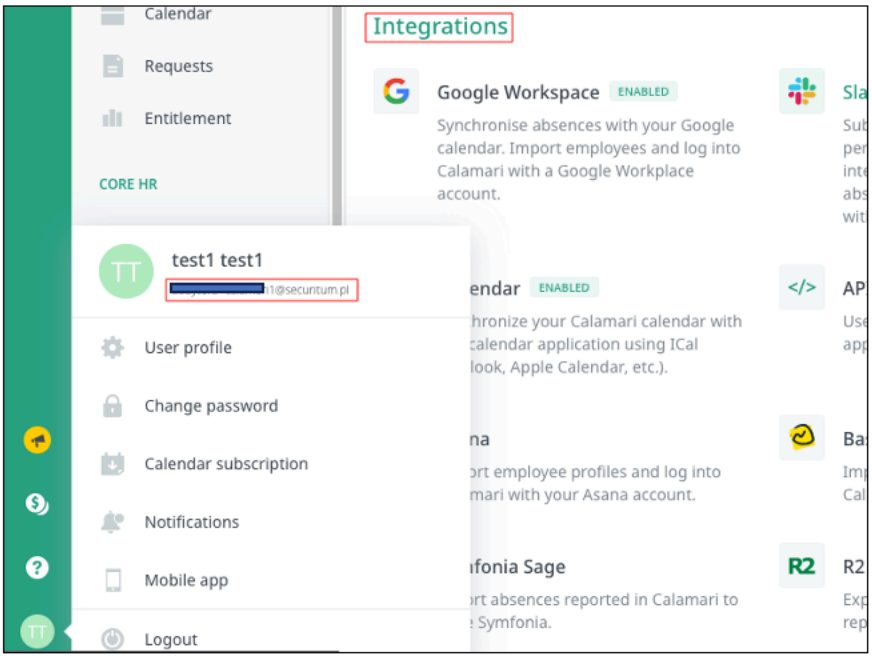
PREREQUISITES FOR THE ATTACK

Possession of the account credentials that were used when setting up the Google Workspace integration.

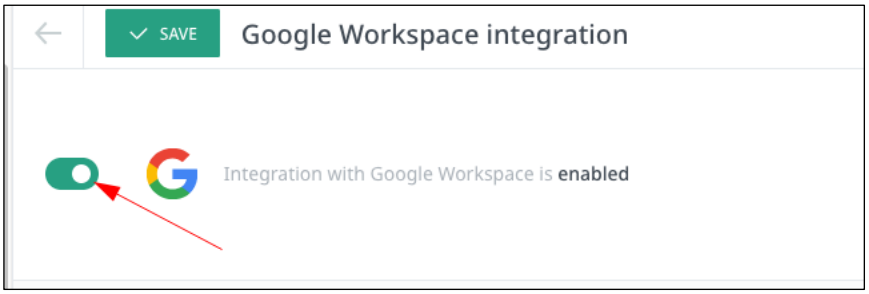
TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to confirm the vulnerability, the following steps have to be taken:

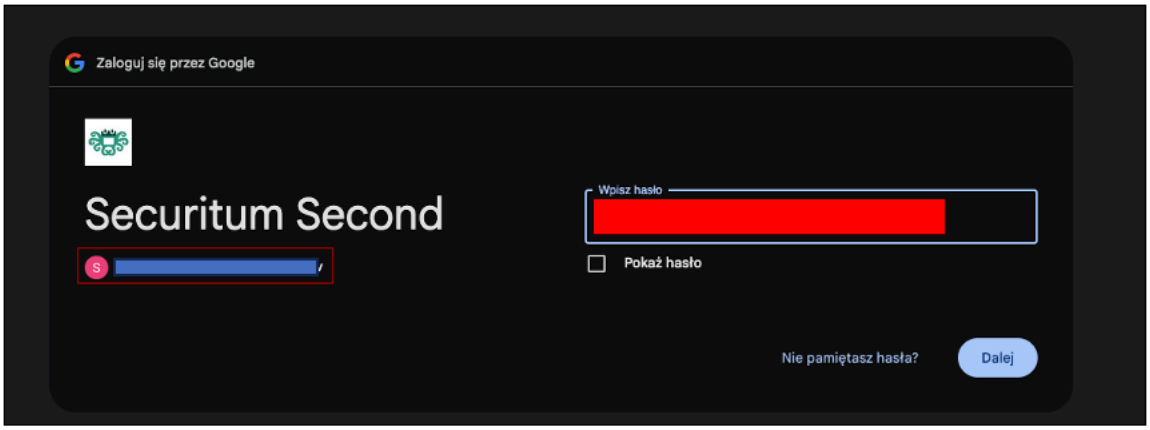
1. Log in to the application as an administrator.
2. Confirm that you are logged in as an administrator:



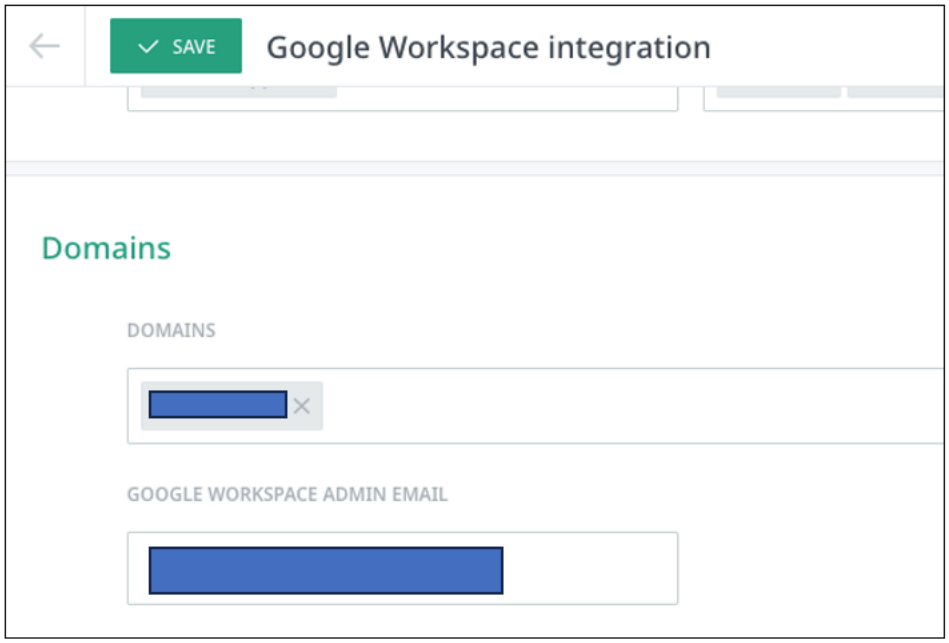
3. Click the Google Workspace integration.
4. Enable the integration:



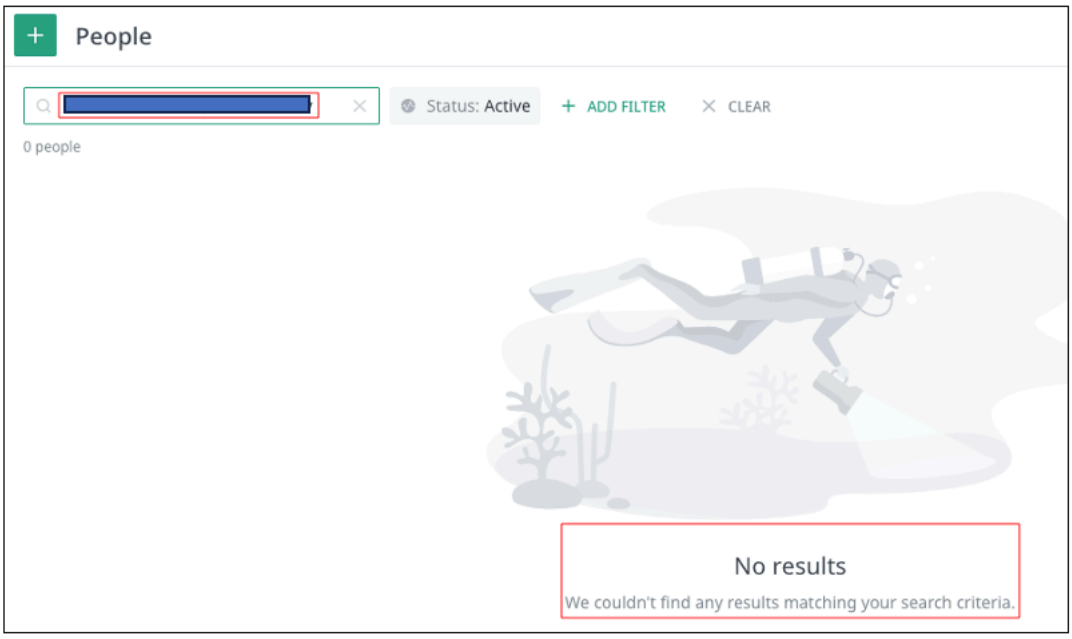
5. Redirection to the Google Account login page will occur.
6. Authentication must be performed with a standard user account (without admin privileges granted in the Google Workspace administration panel):



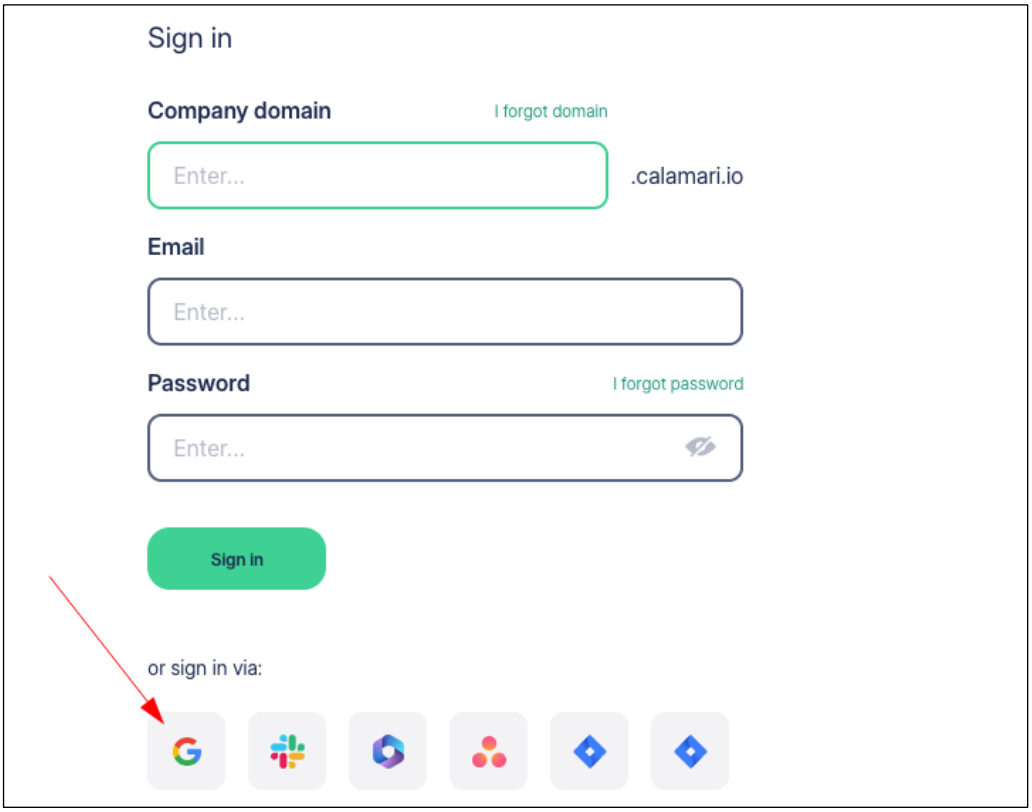
7. Once Google authentication is finalized, the return to the integration setup will occur:



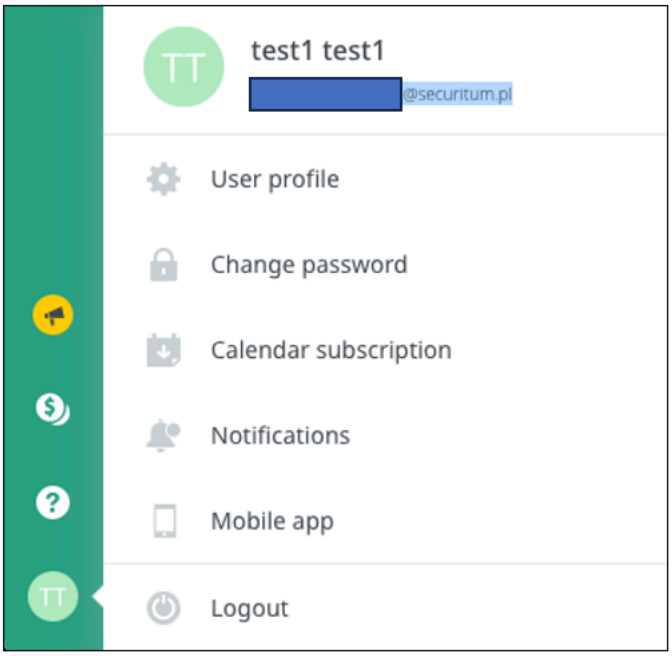
8. Subsequently, confirmation is required that the integration account ([REDACTED]) is not listed among the application users:



9. Log out of the administrator account.
10. On the login page, select the method using Google:



11. An active administrative session is obtained after logging in with the Google account ([REDACTED]). This grants the ability to execute administrative actions while impersonating the account (audytor3+calamari1@securitum.pl):



LOCATION

Google Workspace integration.

RECOMMENDATION

It is recommended to verify the described behaviour and correct its cause. A user used to configure the integration should not be able to access another user's administrative account.

It is also recommended that an account with regular Google Workspace user privileges cannot be used to configure the integration.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V8.2.1 – Verify that the application ensures that function-level access is restricted to consumers with explicit permissions.

[FIXED][LOW] SECURITUM-2418465-016: Redundant information revealed in detailed error messages

STATUS AFTER RETEST (29.04.2026)

The vulnerability has been fixed.

SUMMARY

During the tests, it was observed that the application reveals detailed error messages. An attacker using this fact may learn the application in more detail, including identification of the currently used software (e.g. the framework) and obtain valuable information that will help him or her to profile the application and plan further attacks.

More information:

- https://owasp.org/www-community/Improper_Error_Handling
- https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html

PREREQUISITES FOR THE ATTACK

Access to the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example of a request sent to the application:

```
GET /webapi/\..\..\..\..\..\..\..\..\..\etc\passwd HTTP/2
Host: [DOMAIN]
User-Agent: Mozilla[…]
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/clockin/main-new.do?iframe=true
Content-Length: 0
Te: trailers
```

In response, the application reveals a detailed error message:

```
HTTP/2 400 Bad Request
Date: Wed, 24 Sep 2025 14:31:24 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 2094
Set-Cookie: […]
Content-Language: en

<p><b>Message</b> Invalid character found in the request target
[&#47;webapi&#47;\..\..\..\..\..\..\..\..\..\etc\passwd ]. The valid characters are defined in
RFC 7230 and RFC 3986</p><p><b>Description</b> The server cannot or will not process the request
due to something that is perceived to be a client error (e.g., malformed request syntax, invalid
request message framing, or deceptive request
routing).</p><p><b>Exception</b></p><pre>java.lang.IllegalArgumentException: Invalid character
```

```
found in the request target [&#47;webapi&#47;\..\..\..\..\..\..\..\..\..\etc\passwd ]. The valid
characters are defined in RFC 7230 and RFC 3986
        org.apache.coyote.http11.Http11InputBuffer.parseRequestLine(Http11InputBuffer.java:493)
        org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:269)
        org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:63)
        org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:937)
        org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1793)
        org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:52)
        org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1190)
        org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
        org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
        java.base&#47;java.lang.Thread.run(Thread.java:840)
</pre><p><b>Note</b> The full stack trace of the root cause is available in the server logs.</p><hr
class="line" /><h3>Apache Tomcat/9.0.102</h3>
```

LOCATION

Generic recommendation.

RECOMMENDATION

It is recommended to disable error reporting and replace messages with one consistent with the mapped error identifier without disclosing redundant information.

More information:

- https://owasp.org/www-community/Improper_Error_Handling#how-to-protect-yourself
- https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Error_Handling_Cheat_Sheet.md
- https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V16.5.1 – Verify that a generic message is returned to the consumer when an unexpected or security-sensitive error occurs, ensuring no exposure of sensitive internal system data such as stack traces, queries, secret keys, and tokens.

Informational issues

[NOT VERIFIED][INFO] SECURITUM-2418465-017: Using numerical resource identifiers

SUMMARY

It has been observed that the application uses numerical resource identifiers (e.g., a project with ID 3). This approach in itself does not constitute a security vulnerability. However, in the event of an unauthorized access vulnerability (SECURITUM-2418465-001), the predictability of the identifiers significantly facilitates attack execution. A recommended practice is to use unpredictable identifiers, for example, in the form of UUIDv4.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of using a numerical resource identifier (a project ID):

```
POST /webapi/clockin/workloging/switch-job HTTP/2
Host: [DOMAIN]
Cookie: […]
Content-Length: 15
X-Csrf-Token: […]
Accept-Language: en
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[DOMAIN_2]
Referer: https://[DOMAIN_2]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"projectId":3}
```

LOCATION

A general recommendation.

RECOMMENDATION

It is recommended to use unpredictable resource identifiers, such as UUIDv4.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V11.5.1 – Verify that all random numbers and strings which are intended to be non-guessable must be generated using a cryptographically secure pseudo-random number generator (CSPRNG) and have at least 128 bits of entropy.

[PARTIALLY IMPLEMENTED][INFO] SECURITUM-2418465-018: User enumeration

STATUS AFTER RETEST (03.06.2026)

The information point has been partially implemented and accepted by the Client.

STATUS AFTER RETEST (29.04.2026)

The information point has been partially implemented.

It is still possible to perform the user enumeration:

```
POST /webapi/tenant/signup/with-email HTTP/2
Host: [DOMAIN]
Cookie: […]
Content-Length: 102
Sec-Ch-Ua-Platform: "macOS"
X-Csrf-Token: 96f1f487-6905-470a-bf62-b40a5f779e5f
Accept-Language: pl-pl
Sec-Ch-Ua: "Not-A.Brand";v="24", "Chromium";v="146"
Sec-Ch-Ua-Mobile: ?0
User-Agent: […]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[DOMAIN]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[DOMAIN]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive

{"contactUserEmail":"[…]@securitum.pl","agreement":true,"acquisitionSourceCampaign":null}
```

Server response:

```
HTTP/2 409 Conflict
Date: Tue, 28 Apr 2026 11:56:40 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 101
Set-Cookie:
AWSALB=qbFV2LQ1SSti2QAN8JWxswzs+RmuD0AIrQEQAuozQZsR19ZHUbhnMbYemRWcdRCLQifjGM4jDPA7uE+PzqfCZaYT8u
Tb2ZbtDN+6omi2y/xvDSeDFKnlftX1PlbC; Expires=Tue, 05 May 2026 11:56:40 GMT; Path=/
Set-Cookie:
AWSALBCORS=qbFV2LQ1SSti2QAN8JWxswzs+RmuD0AIrQEQAuozQZsR19ZHUbhnMbYemRWcdRCLQifjGM4jDPA7uE+PzqfCZa
YT8uTb2ZbtDN+6omi2y/xvDSeDFKnlftX1PlbC; Expires=Tue, 05 May 2026 11:56:40 GMT; Path=/;
SameSite=None; Secure
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
```

```
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server:
Strict-Transport-Security: max-age=31536000

{"fields":{"contactUserEmail":"Ten e-mail został już zarejestrowany"},"nonfields":[],"warnings":[]}
```

Additionally, it is still possible to perform the enumeration on the registration endpoint due to the subtle response difference.

Example request (registered account):

```
GET /webapi/tenant/signup/email-status?email=audytor5%40securitum.pl HTTP/2
Host: [DOMAIN]
Sec-Ch-Ua-Platform: "macOS"
X-Csrf-Token: 96f1f487-6905-470a-bf62-b40a5f779e5f
Accept-Language: pl-pl
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Not-A.Brand";v="24", "Chromium";v="146"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/146.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Origin: https://[DOMAIN]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[DOMAIN]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

Example response:

```
HTTP/2 200 OK
Date: Tue, 28 Apr 2026 08:10:56 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 37
Set-Cookie: AWSALB=GImoQM/GU0vvcYixO8/g0/xOUUQR0EwKroT[…]; Expires=Tue, 05 May 2026 08:10:56 GMT;
Path=/
Set-Cookie: AWSALBCORS=GImoQM/GU0vvcYixO8/g0/xOUUQ[…]; Expires=Tue, 05 May 2026 08:10:56 GMT;
Path=/; SameSite=None; Secure
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Set-Cookie: _csrf_token=658a8916-93fc-4fa0-8319-67eec799916a; Max-Age=2592000; Expires=Thu, 28 May
2026 08:10:56 GMT; Domain=calamari.io; Path=/; Secure; SameSite=None
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
```

```
Server:
Strict-Transport-Security: max-age=31536000

{"status":"EMAIL_ALREADY_REGISTERED"}
```

Example request (non-registered email):

```
GET /webapi/tenant/signup/email-status?email=[…]%40securitum.pl HTTP/2
Host: [DOMAIN]
Sec-Ch-Ua-Platform: "macOS"
X-Csrf-Token: 96f1f487-6905-470a-bf62-b40a5f779e5f
Accept-Language: pl-pl
Accept: application/json, text/plain, */*
Sec-Ch-Ua: "Not-A.Brand";v="24", "Chromium";v="146"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/146.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Origin: https://[DOMAIN]
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://[DOMAIN]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

Example response:

```
HTTP/2 200 OK
Date: Tue, 28 Apr 2026 08:11:24 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 48
Set-Cookie: AWSALB=XgkER4Mof5XdyC4wlBp4Zn2pvF[…]; Expires=Tue, 05 May 2026 08:11:24 GMT; Path=/
Set-Cookie: AWSALBCORS=XgkER4Mof5XdyC4wlBp[…]; Expires=Tue, 05 May 2026 08:11:24 GMT; Path=/;
SameSite=None; Secure
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Set-Cookie: _csrf_token=b5a0dac9-06c1-4ff7-ae8c-65fe159e3ab0; Max-Age=2592000; Expires=Thu, 28 May
2026 08:11:24 GMT; Domain=calamari.io; Path=/; Secure; SameSite=None
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server:
Strict-Transport-Security: max-age=31536000

{"status":"ALREADY_REGISTERED_FROM_THIS_DOMAIN"}
```

SUMMARY

During the tests, it was discovered that it is possible to enumerate existing users in the system. This can be helpful in further attacks, if their aim is to perform login attempts to the application.

More information:

- https://wiki.owasp.org/index.php/Testing_for_User_Enumeration_and_Guessable_User_Account_(OWASP-AT-002)

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – Login form

Enumerating users requires following the steps described below:

1. Go to the login form.
2. Complete it and click "Sign in".
3. Sending the following HTTP request to the application makes it possible to discover existing users by sequentially changing the user identifier (highlighted in yellow):

```
POST /sign-in.do HTTP/2
Host: [DOMAIN]
Cookie: […]
Content-Length: 87
X-Csrf-Token: […]
Accept-Language: en
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"domain":"securitum2025","login":"[...]@securitum.pl","password":"test"}
```

4. If the user exists (and it is blocked), the following message is returned in response:

```
HTTP/2 401 Unauthorized
Date: Wed, 24 Sep 2025 10:16:08 GMT
Content-Length: 67
[…]
X-Content-Type-Options: nosniff

{"success":false,"response":"loginTemporaryLocked","redirect":null}
```

5. On the other hand, if no match is found (e.g. test@securitum.pl) other message is returned:

```
HTTP/2 401 Unauthorized
Date: Wed, 24 Sep 2025 10:20:12 GMT
Content-Length: 57
[…]
X-Content-Type-Options: nosniff

{"success":false,"response":"loginerror","redirect":null}
```

Example #2 – Registration form

Enumerating users requires following the steps described below:

1. Go to the registration form (https://[DOMAIN]/sign-up).
2. Click "Start as company".

3. Insert email address e.g., audytor6+calamari01@securitum.pl and click “Start free trial”.
4. Sending the following HTTP request to the application makes it possible to discover existing users by sequentially changing the user identifier (highlighted in yellow):

```
POST /webapi/tenant/signup/with-email HTTP/2
Host: [DOMAIN]
Cookie: […]
Content-Length: 105
X-Csrf-Token: […]
Accept-Language: en
User-Agent: Mozilla[…]
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://[DOMAIN]
Referer: https://[DOMAIN]/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"contactUserEmail":"[…]@securitum.pl","agreement":true,"acquisitionSourceCampaign":null}
```

5. If the user exists, the following message is returned in response:

```
HTTP/2 409 Conflict
Date: Tue, 30 Sep 2025 11:33:44 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 101
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

{"fields":{"contactUserEmail":"This email has already been
registered"},"nonfields":[],"warnings":[]}
```

6. On the other hand, if the user has not been previously registered, a different message is returned:

```
HTTP/2 200 OK
Date: Tue, 30 Sep 2025 11:59:58 GMT
Content-Type: application/json;charset=UTF-8
Content-Length: 16
Set-Cookie: AWSALB=[…]; Expires=Tue, 07 Oct 2025 11:59:58 GMT; Path=/
Set-Cookie: […]; Expires=Tue, 07 Oct 2025 11:59:58 GMT; Path=/; SameSite=None; Secure
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: https://[DOMAIN]
Access-Control-Expose-Headers: Content-Disposition
Access-Control-Allow-Credentials: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

```
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

{"result": true}
```

LOCATION

Example #1

- https://[DOMAIN]/sign-in.do

Example #2

- https://[DOMAIN]/webapi/tenant/signup/with-email

RECOMMENDATION

It is recommended to create a unified message, e.g. "*Invalid email or password.*". Even if the account has already been blocked.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V6.3.8 – Verify that valid users cannot be deduced from failed authentication challenges, such as by basing on error messages, HTTP response codes, or different response times. Registration and forgot password functionality must also have this protection.

[NOT VERIFIED][INFO] SECURITUM-2418465-019: X-XSS-Protection header enabled

SUMMARY

It was observed that HTTP responses contain `X-XSS-Protection` header. This header is not supported anymore by majority of the browsers (such as Chrome, Mozilla Firefox, Microsoft Edge), and in very rare and specific cases may open an application to the XS-Leak vulnerability. The role of `X-XSS-Protection` header was taken over by a Content Security Policy.

More information:

- https://markitzeroday.com/headers/content-security-policy/2018/02/10/x-xss-protection.html
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection
- https://portswigger.net/daily-swig/google-deprecates-xss-auditor-for-chrome

LOCATION

https://[DOMAIN]/

RECOMMENDATION

It is recommended to verify if the `X-XSS-Protection` header is necessary (for example, if an application is used in very old browsers, which do not support Content Security Policy). If not, it should be deleted.

It should be noted that its occurrence does not automatically open an application to new vulnerabilities (as the exploitation of XS-Leaks may be very sophisticated and not possible in each case), and this recommendation is only a suggestion, allowing for additional hardening.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.4 – Browser Security Mechanism Headers

[NOT VERIFIED][INFO] SECURITUM-2418465-020: Lack of Referrer-Policy header

SUMMARY

It was identified that the tested application does not implement `Referrer-Policy` header.

This header allows to specify what information can be placed in the `Referer` request header. It is also possible to disable sending any values in the `Referer` header which will prevent from leaking sensitive information to other third-party servers.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy
- https://scotthelme.co.uk/a-new-security-header-referrer-policy/

LOCATION

https://[DOMAIN]/

RECOMMENDATION

`Referrer-Policy` header should be added in all server responses:

```
Referrer-Policy: [value]
```

where `[value]` should have one of the following values:

- no-referrer: `Referer` header will never be sent in the requests to server.
- origin: `Referer` header will be set to the origin from which the request was made.
- origin-when-cross-origin: `Referer` header will be set to the full URL in requests to the same origin but only set to the origin when requests are cross-origin.
- same-origin: `Referer` header contains full URL for requests to the same origin, in other requests the `Referer` header is not sent.

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.4.5 – Verify that the application sets a referrer policy to prevent leakage of technically sensitive data to third-party services via the 'Referer'http request header field. This can be done using the Referrer-Policy http response header field or via HTML element attributes. Sensitive data could include path and query data in the URL, and for internal non-public applications also the hostname.

[IMPLEMENTED][INFO] SECURITUM-2418465-021: Host Header Injection

STATUS AFTER RETEST (29.04.2026)

The information point has been implemented.

SUMMARY

During the tests, a possibility of injecting own value into the `Host` header was detected.

The vulnerability was assessed at the "*General recommendations*" level, due to the inability to remotely inject the `Host` header into the request.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example request sent to the server with `Host` header value changed (HTTP/1.1, port 80, server: [DOMAIN]):

```
GET / HTTP/1.1
Host: rto7kzsqg7vukdqixw8wycnx5obfz7nw.x.securak.pl
```

In response, the server returns the HTTP 302 code with the address entered in the `Host` header:

```
HTTP/1.1 301 Moved Permanently
Server: awselb/2.0
Date: Wed, 24 Sep 2025 12:04:08 GMT
Content-Type: text/html
Content-Length: 134
Connection: keep-alive
Location: https://rto7kzsqg7vu[...].x.[...].pl:443/

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
```

LOCATION

https://[DOMAIN]

RECOMMENDATION

It is recommended to filter the data received from the user and carefully verify that the `Host` header value matches the expected domain (e.g. application address).

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V2.2.1 – Verify that input is validated to enforce business or functional expectations for that input. This should either use positive validation against an allow list of values, patterns, and ranges, or be based on comparing the input to an expected structure and logical limits according to predefined rules.

[IMPLEMENTED][INFO] SECURITUM-2418465-022: Lack of general field validation

STATUS AFTER RETEST (29.04.2026)

The information point has been implemented.

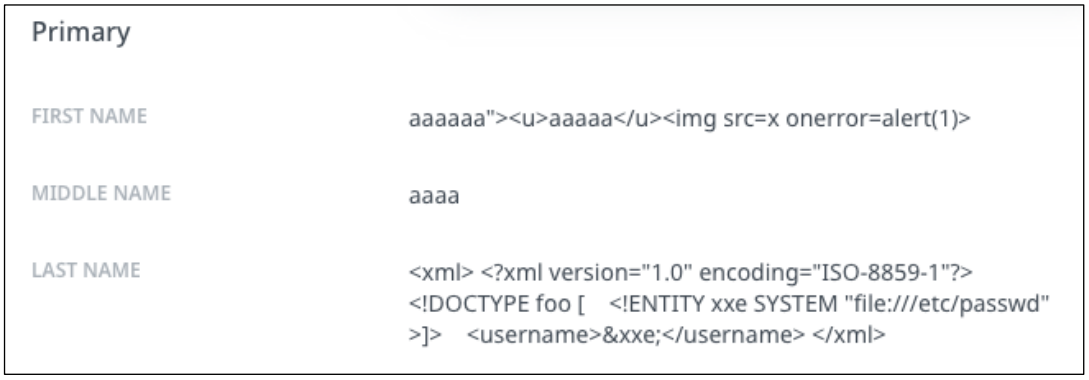
SUMMARY

During the test, it was observed that most of the fields do not have the correctly implemented server-side verification, currently there is only a partial client-side verification.

Point does not currently create a security risk, but if the application communicates with other systems, it may cause unexpected behaviour.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below is an example screenshot from the application:



LOCATION

Generic recommendation.

RECOMMENDATION

It is recommended to validate all the data received from a user (rejection of values inconsistent with the template/format of a given field – whitelist approach), and then encode it on the output in relation to the context in which it is embedded (in all places of application, not only those specified in the description).

For this purpose, it should be verified whether the framework used by the application has built-in functions that implement the described recommendation.

It is worth noting that the server should not return the values that caused the error but only indicate that: "*An incorrect value was entered in the XYZ field*", alternatively with additional information about allowed characters, e.g. "*Allowed characters are letters and numbers*".

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V2.2.1 – Verify that input is validated to enforce business or functional expectations for that input. This should either use positive validation against an allow list of values, patterns, and ranges, or be based on comparing the input to an expected structure and logical limits according to predefined rules.

[NOT VERIFIED][INFO] SECURITUM-2418465-023: Lack of Content-Disposition header

SUMMARY

From a security perspective, it is a good practice to add a `Content-Disposition` header to the HTTP API response, which will force the web browser not to interpret the response content under any circumstances. Forcing such behavior on the application may protect the application against vulnerabilities, including for Cross-Site Scripting (XSS).

LOCATION

https://[DOMAIN]/

RECOMMENDATION

`Content-Disposition` header should be added in all server responses:

```
Content-Disposition: attachment; filename="api.json"
```

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.2.1 – Verify that security controls are in place to prevent browsers from rendering content or functionality in HTTP responses in an incorrect context (e.g., when an API, a user-uploaded file or other resource is requested directly). Possible controls could include: not serving the content unless HTTP request header fields (such as Sec-Fetch-*) indicate it is the correct context, using the sandbox directive of the Content-Security-Policy header field or using the attachment disposition type in the Content-Disposition header field.

[PARTIALLY IMPLEMENTED][INFO] SECURITUM-2418465-024: Lack of Content-Security-Policy header

STATUS AFTER RETEST (03.06.2026)

The information point has been partially implemented. However, the Client confirmed that indicated endpoints will be removed soon.

STATUS AFTER RETEST (29.04.2026)

The information point has not been implemented.

Example request:

```
POST /absence/absence/calendar.do HTTP/2
Host: [DOMAIN]
Cookie: […]
[…]
```

Example response:

```
HTTP/2 200 OK
Date: Mon, 27 Apr 2026 14:02:31 GMT
Content-Type: text/html;charset=UTF-8
Content-Length: 1518
Set-Cookie: […]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Initable: true
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Language: pl-PL
Server:
Strict-Transport-Security: max-age=31536000
```

Example request #2:

```
GET /absence/main-new.do?iframe=true HTTP/2
Host: [DOMAIN]
Cookie: […]
[…]
```

Example response #2:

```
HTTP/2 200 OK
Date: Mon, 27 Apr 2026 14:02:27 GMT
Content-Type: text/html;charset=UTF-8
Content-Length: 1490
Set-Cookie: […]
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

```
Accept-Ranges: bytes
Etag: W/"1490-1777274454000"
Last-Modified: Mon, 27 Apr 2026 07:20:54 GMT
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Language: pl-PL
Server:
Strict-Transport-Security: max-age=31536000
```

SUMMARY

The `Content-Security-Policy` (CSP) header was not identified in the application responses.

Content Security Policy is a security mechanism operating at the browser level that aims to protect it against the effects of vulnerabilities acting on the browser side (e.g. Cross-Site Scripting). CSP may significantly impede the exploitation of vulnerabilities, however its implementation may be complicated and may require significant changes in the application structure.

The main idea of CSP is to define a list of allowed sources from which external resources can be loaded on the page. For example, if you define the following CSP policy:

```
Content-Security-Policy: default-src 'self'
```

all external resources on the webpage may be loaded only from the application's domain (`'self'`), and due to that, any attempt to load script or image from external domain will fail. In this implementation, it is also impossible to define the script code directly in the HTML code, e.g.:

```
<script>jQuery.ajax(...)</script>
```

All scripts must be defined in external files, e.g.:

```
<script src="/app.js"></script>
```

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

LOCATION

https://[DOMAIN]/

RECOMMENDATION

It is recommended to consider implementation of the `Content-Security-Policy` header. To do this, define all domains from which the resources in the application are downloaded (images, scripts, video/audio elements, CSS styles etc.) and build CSP policy based on them.

If a large number of scripts defined directly in the HTML code (`<script>` tags or events such as `onclick`) is used, they should be placed in external JavaScript files or `nonce` policies should be used. More information is included in the links below:

- https://csp-evaluator.withgoogle.com/
- https://csp.withgoogle.com/docs/index.html

- https://report-uri.com/home/generate

UNFULFILLED ASVS 5.0 L2 REQUIREMENTS

V3.4.3 – Verify that HTTP responses include a Content-Security-Policy response header field which defines directives to ensure the browser only loads and executes trusted content or resources, in order to limit execution of malicious JavaScript. As a minimum, a global policy must be used which includes the directives object-src 'none' and base-uri 'none' and defines either an allowlist or uses nonces or hashes. For an L3 application, a per-response policy with nonces or hashes must be defined.