securitum

Security report

SUBJECT

Client_Name web application

DATE

31.07.2024 – 08.08.2024

RETEST DATE

29.01.2025

LOCATION

Cracow

AUTHOR

Bartosz Dyczkowski

VERSION

1.1

securitum

Executive summary

This document is a summary of work conducted by the Securitum. The subject of the test was the Client_Name web application .

Tests were conducted using the following roles:

- Dyrektor regionalny / Regional Director
- Super Admin
- Użytkownik nieuwierzytelniony / Unauthenticated User

The most significant vulnerability identified is:

- [HIGH] SECURITUM-XXXXXX-001: Authorization flaws - possibility of accessing unauthorized functionalities.

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 the Securitum.

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

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

Status after retests 29.01.2025

On January 29, 2025, retests were conducted for all previously identified vulnerabilities and informational points.

As a result of this work, a “Retest status” section was added to each verified vulnerability or recommendation, containing detailed information about the retesting process.

A total of 7 vulnerabilities and 5 informational points were retested.

The retesting was carried out in the staging environment available at: https://[DOMAIN]

Below is a table presenting the vulnerability name, its classification, and status.

Vulnerability ID	Vulnerability or recommendation title	Status after retests
SECURITUM-XXXXXX-001	Authorization flaws - possibility of accessing unauthorized functionalities	Fixed
SECURITUM-XXXXXX-002	Lack of protection against brute force attack on login form	Fixed
SECURITUM-XXXXXX-003	Support for HTTP (unencrypted) communication	Fixed
SECURITUM-XXXXXX-004	Open Redirect – possibility to redirect a user to a malicious domain	Fixed
SECURITUM-XXXXXX-005	Lack of validation for files within archives uploaded to the server	Fixed
SECURITUM-XXXXXX-006	Path traversal - possibility to upload a file outside the designated directory	Fixed
SECURITUM-XXXXXX-007	Authentication via HTTP Basic Authentication	Fixed
SECURITUM-XXXXXX-008	Lack of Content-Security-Policy header	Not implemented
SECURITUM-XXXXXX-009	Lack of Referrer-Policy header	Not implemented
SECURITUM-XXXXXX-010	Lack of Strict-Transport-Security (HSTS) header	Not implemented
SECURITUM-XXXXXX-011	Lack of X-Content-Type-Options header	Not implemented
SECURITUM-XXXXXX-012	No invalidation of the session after logout	Not 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.

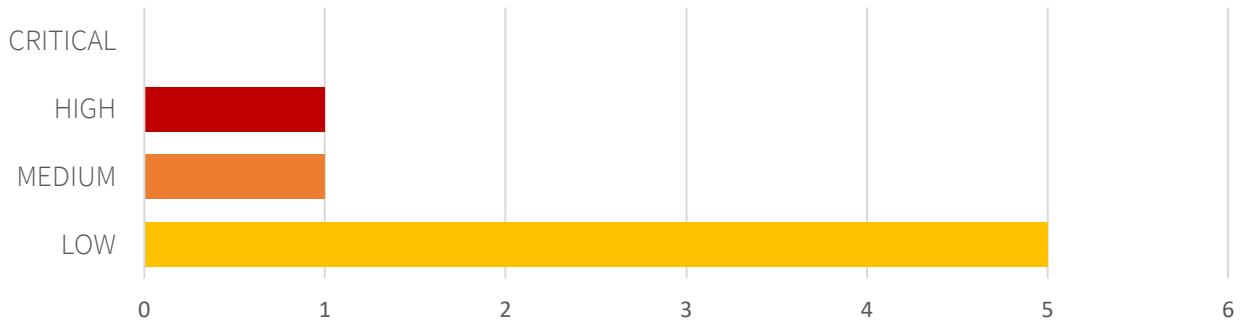
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser 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, 5 INFO issues are reported.

Statistical overview following the retest - 29 January 2025

All vulnerabilities were fixed.

Additionally, 5 informational issues remain present.

Contents

Change history

Document date	Version	Change description
30.01.2025	1.1	Creation of retest report. A re-verification of all previously reported findings was conducted.
08.08.2024	1.0	Creation of the security testing report.

Vulnerabilities in the web application

[FIXED] [HIGH] SECURITUM-XXXXXX-001: Authorization flaws - possibility of accessing unauthorized functionalities

RETEST STATUS 29.01.2025

The vulnerability has been fixed.

SUMMARY

The tested application does not implement a proper authorization layer for certain API endpoints. As a result, an application user can perform actions that are not available or visible through the graphical user interface.

For example, exploiting this vulnerability allows:

- Deletion of a user from the „Web Panel”
- Modification of an existing notification

More information:

- 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

An account in the application with „Super Admin” permissions.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example 1: Deleting a user from the „Web panel”

A user with "Super Admin" privileges is only allowed to view user details in the „Web panel” section.

Deleting a user from the „Web panel” requires the following steps:

1. Authenticate in the application using an account with „Super Admin” privileges.
2. Go to the details of the selected account to obtain the necessary ID:

3. Send the following HTTP request to the application:

```
DELETE /web-panel/users/mobile/07bf75b1-9be4-[…] HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […REDACTED…]
```

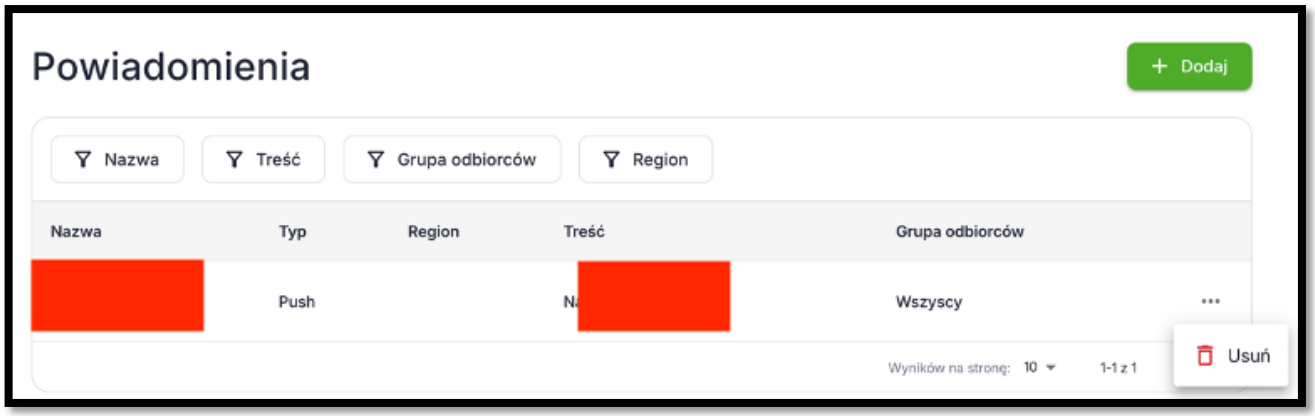
Server response confirming the deletion of the user:

```
HTTP/2 204 No Content
Date: Wed, 07 Aug 2024 10:27:54 GMT
Server: Kestrel
Access-Control-Allow-Origin: *
```

4. Vulnerability confirmation:

Example 2: Modifying an existing notification

A user with „Super Admin" privileges is only allowed to delete notifications and create new ones:



Modifying an existing notification requires the following steps:

1. Authenticate in the application using an account with „Super Admin" privileges.
2. Navigate to the „Powiadomienia (Notifications)" section - the application sends a request to retrieve notification information, and the response allows extraction of the notification ID:

```
GET /web-panel/notifications?PageNumber=1&PageSize=10 HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
```

Server response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

{
  "items" : [
    {
      "id" : "94f12ce5-0109-[…]",
      "name" : "Test <s>aa",
      "message" : "Test<s> {{7*7}}",
      "type" : "Push",
      "receivers" : "All",
      "districts" : [ ],
      "createdAt" : "2024-08-01T12:01:40.087711Z"
    }
  ],
  "empty" : false,
  "currentPage" : 1,
  "resultsPerPage" : 10,
  "totalPages" : 1,
  "totalResults" : 1
}
```

3. Send the following HTTP request to the application:

```
PUT /web-panel/notifications/94f12ce5-0109-[…] HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
Content-Type: application/json
Content-Length: 141

{
  "id" : "94f12ce5-0109-[…]",
  "name" : "Zmodyfikowane",
  "message" : "po czasie",
  "type" : "Push",
  "receivers" : "All",
  "districtIDs" : null
}
```

Server response confirming the modification of the notification:

```
HTTP/2 204 No Content
Date: Wed, 07 Aug 2024 11:07:43 GMT
Server: Kestrel
Access-Control-Allow-Origin: *
```

4. Vulnerability confirmation:

LOCATION

Example 1: Deleting a user from the „Web panel”:

- [HOSTNAME]/web-panel/users/mobile/{id} – DELETE HTTP method.

Example 2: Modifying an existing notification:

- [HOSTNAME]/web-panel/notifications/{id} - PUT HTTP method.

RECOMMENDATION

It is recommended to improve the mechanism responsible for verifying permissions for the use of specific methods in requests sent to API endpoints. A user should only be able to perform actions explicitly available in the application's GUI.

More information:

- 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

[FIXED] [MEDIUM] SECURITUM-XXXXXX-002: Lack of protection against brute force attack on login form

RETEST STATUS 29.01.2025

The vulnerability has been fixed. A user account is blocked after several failed login attempts. However, it should be noted that a potential attacker can intentionally block user accounts if the login names are known.

SUMMARY

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

More information:

- https://owasp.org/www-community/attacks/Brute_force_attack

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

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

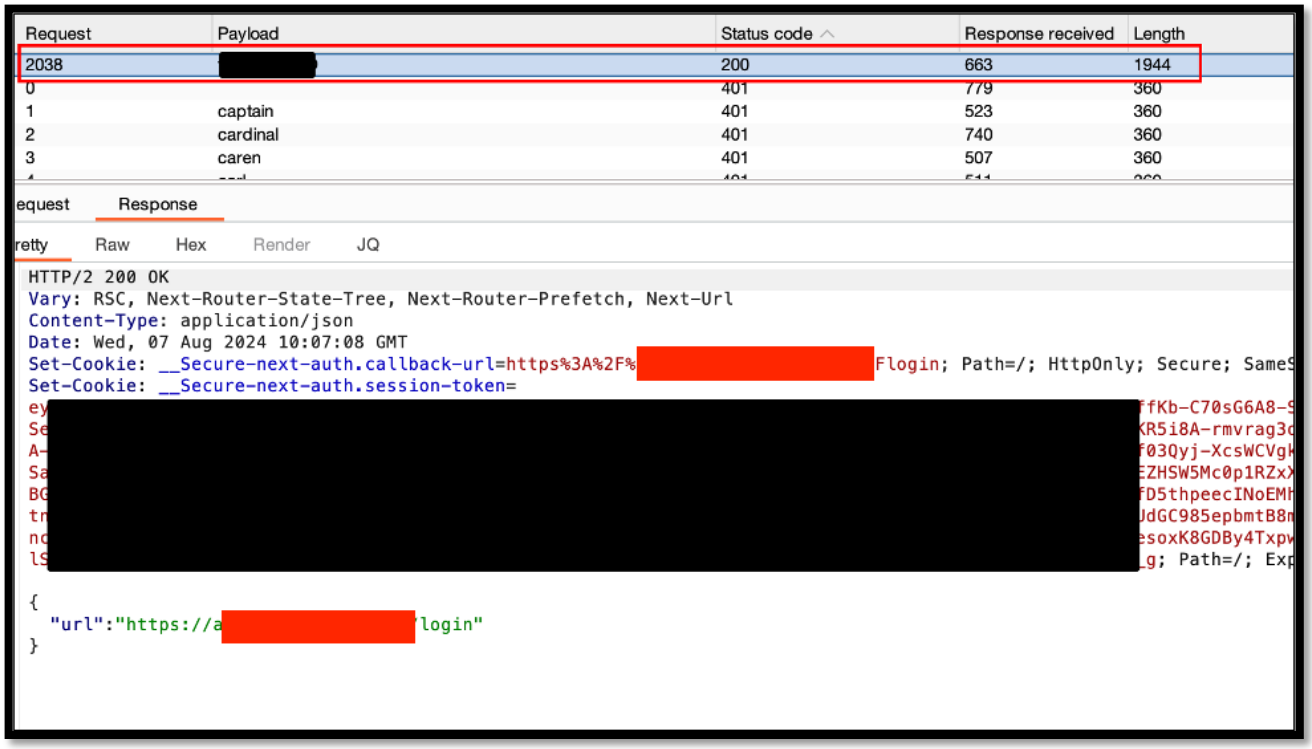
1. Navigate to the application's login form at [HOSTNAME]/login.
2. Enter the username, e.g., „audytor11+Client_Name01".
3. The attacker initiates enumeration by entering a potential user password.
4. Enumeration continues as the attacker submits additional password combinations.

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 2037 failed attempts made to log into the test account with an incorrect password (highlighted in yellow):

```
POST /api/auth/callback/credentials HTTP/2
Host: [HOSTNAME]
Cookie: NEXT_LOCALE=pl; __Host-next-auth.csrf-token=[…]; __Secure-next-auth.callback-
url=https%3A%2F%2F[HOSTNAME]%2Fa[…]s
Content-Type: application/x-www-form-urlencoded
Content-Length: 207
Connection: keep-alive

redirect=false&username=audytor11%2BClient_Name01&password=TestTest&csrfToken=[…]&callbackUrl=htt
ps%3A%2F%2F[HOSTNAME]%2Flogin&json=true
```

The next, 2038 request sent confirms that the account has not been blocked and presents the ending of enumeration with success (correct finding of the password):



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

Login mechanism:

- [HOSTNAME]/api/auth/callback/credentials

RECOMMENDATION

It is recommended that the application blocks mass login attempts either to a single account using multiple passwords or to multiple accounts using the same password. This can be achieved, for example, by implementing CAPTCHA mechanisms or SMS/email tokens when an attack attempt is detected.

More information:

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

[FIXED] [LOW] SECURITUM-XXXXXX-003: Support for HTTP (unencrypted) communication

RETEST STATUS 29.01.2025

The vulnerability has been fixed. The ability to communicate using an unencrypted protocol has been blocked.

SUMMARY

Application testing revealed that the application does not enforce the use of the encrypted HTTPS protocol.

This creates a risk of sensitive user data being intercepted or modified if an attacker eavesdrops on network traffic („Man in the Middle", MiTM).

More information:

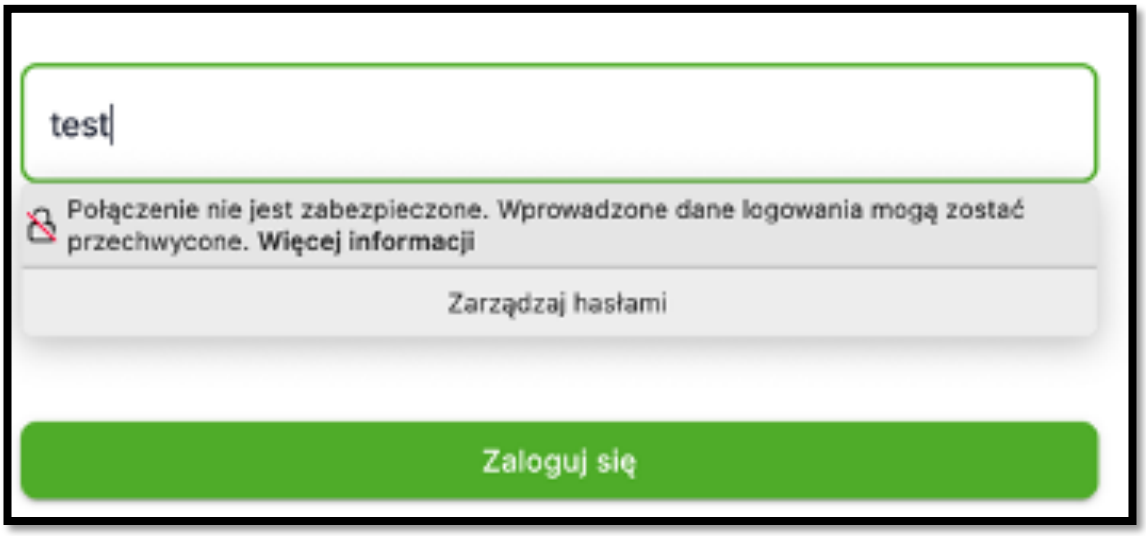
- https://sekurak.pl/kilka-slow-o-wdrozeniu-ssl-i-tls-cz-i/
- https://sekurak.pl/kilka-slow-o-wdrozeniu-ssl-i-tls-cz-ii/
- https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html
- https://cwe.mitre.org/data/definitions/757.html
- https://cwe.mitre.org/data/definitions/326.html

PREREQUISITES FOR THE ATTACK

Conducting a Man-in-the-Middle (MiTM) attack.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The server hosting the tested application allows the use of the unencrypted HTTP protocol and does not enforce the use of the encrypted HTTPS protocol:



LOCATION

Server configuration:

- http://[HOSTNAME]

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

[FIXED] [LOW] SECURITUM-XXXXXX-004: Open Redirect – possibility to redirect a user to a malicious domain

RETEST STATUS 29.01.2025

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

Logging in by link prepared by attackers.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The application allows setting a redirect address after a successful login. An attacker can craft a link that, upon login, redirects the victim to a malicious site specified in the `callbackURL` parameter.

Example links sent to the victim:

- [HOSTNAME]/login?callbackUrl=%2F%2F5fltdiv9c[…]6fc60xom.attacker.domain
- [HOSTNAME]/login?callbackUrl=https://8z8wxlf[…]w9ka8z.attacker.domain

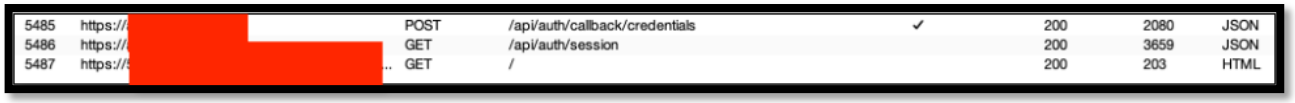
Sending an HTTP login request:

```
POST /api/auth/callback/credentials HTTP/2
Host: [HOSTNAME]
Cookie: __Host-next-auth.csrf-token=[…]
Content-Type: application/x-www-form-urlencoded
Content-Length: 281

redirect=false&username=audytor11%2BClient_Name01&password=[…]&csrfToken=[…]&callbackUrl=https%3A
%2F%2F[HOSTNAME]%2Flogin%3FcallbackUrl%3D%252F%252F5fltdiv9c[…]o6fc60xom.attacker.domain&json=tru
e
```

Server response:

```
HTTP/2 200 OK
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url
Content-Type: application/json
[…]

{ "url":"[HOSTNAME]/login?callbackUrl=%2F%2F5f[…]rfzo6fc60xom.attacker.domain"
}
```

Request history after successful login:



5485	https://	POST	/api/auth/callback/credentials	✓	200	2080	JSON
5486	https://	GET	/api/auth/session		200	3659	JSON
5487	https://	GET	/		200	203	HTML

LOCATION

Login mechanism - `callbackUrl` parameter:

- [HOSTNAME]/login?callbackUrl={ADRES_URL}

RECOMMENDATION

It is recommended to validate the destination address used for redirection, for example by implementing a whitelist of allowed URLs to which users can be redirected.

More information:

- https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.md

[FIXED] [LOW] SECURITUM-XXXXXX-005: Lack of validation for files within archives uploaded to the server

RETEST STATUS 29.01.2025

The vulnerability has been fixed. Uploading zip archives is not possible.

SUMMARY

During testing, it was observed that the application does not validate the contents of uploaded archives. An attacker can exploit this to upload arbitrary files to the server, including executable files or even phishing pages.

In addition, no antivirus software was detected that would scan uploaded files for malicious code.

Important! This vulnerability is global, i.e. it affects all functionalities that allow uploading archive files to the server.

More information:

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.eicar.org/download-anti-malware-testfile/

PREREQUISITES FOR THE ATTACK

Having an account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below is an example request containing a test file (eicar.exe) compressed in a `.zip` archive, which is commonly recognized by antivirus software as potentially dangerous:

```
POST /web-panel/files?fileType=Archive HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
Content-Type: multipart/form-data; boundary--------------------------
672150920416423446024669844l7
Content-Length: 461

------------------------------672150920416423446024669844l7
Content-Disposition: form-data; name="file"; filename="virus.zip"
Content-Type: application/zip

[EICAR_FILE_CONTENT.EXE_IN_ZIP_FORMAT]
------------------------------672150920416423446024669844l7--
```

In response, the server returns a confirmation of the file upload:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

"d3056c1e-e551-[…]"
```

The following request allows downloading of the aforementioned file:

```
GET /abe9ad52-ebfa-[…]/archive/d3056c1e-[…]-12b6d662723c_virus.zip?sv=2023-11-03&se=2024-08-07T15%3A05%3A47Z&sr=b&sp=r&sig=YYV[…]qO4%3D HTTP/1.1
Host: […]
```

In response, the server returns the uploaded file with unchanged content and extension:

```
HTTP/1.1 200 OK
Content-Length: 237
Content-Type: application/zip
[…]

[EICAR_FILE_CONTENT.EXE_IN_ZIP_FORMAT]
```

Contents of the downloaded archive:

```
[…] Downloads % unzip -l d3056c1e-[…]-12b6d662723c_virus.zip
Archive:  d3056c1e-[…]-12b6d662723c_virus.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
       69  01-18-2024 14:21   eicar.exe
---------                     -------
       69                     1 file
```

LOCATION

File upload mechanism:

- [HOSTNAME]/web-panel/files?fileType=Archive

RECOMMENDATION

Every file, including those within an archive, should be validated and scanned for malicious content across all functionalities that involve uploading files to the server.

More information:

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

[FIXED] [LOW] SECURITUM-XXXXXX-006: Path traversal - possibility to upload a file outside the designated directory

RETEST STATUS 29.01.2025

The vulnerability has been fixed. File names containing the parent-directory escape sequence are blocked.

SUMMARY

During testing it was observed that the application allows uploading files to directories outside the defined structure (the path configured in the application). An attacker can exploit this to upload a file to any directory in the container.

More information:

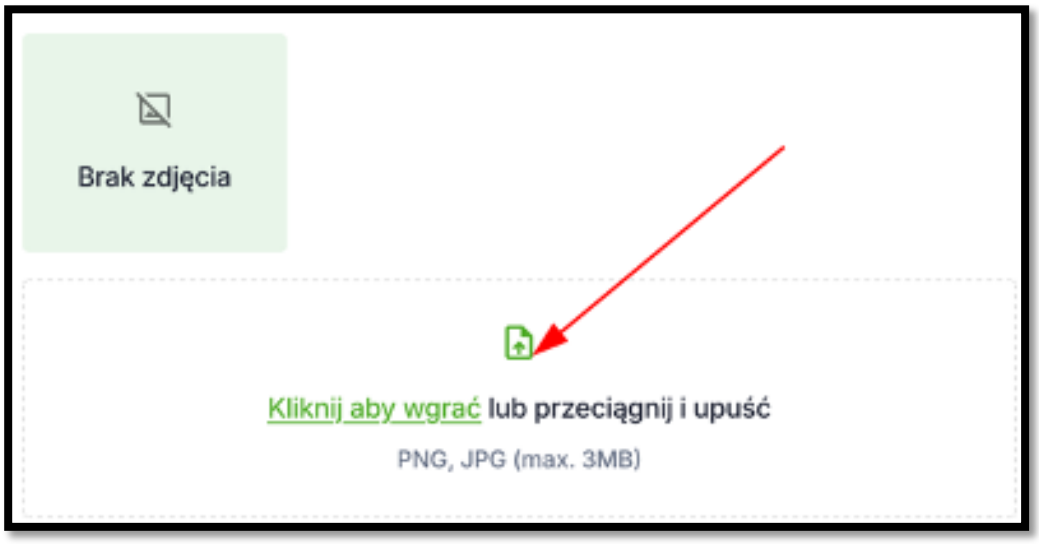
- https://owasp.org/www-community/attacks/Path_Traversal
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/01-Testing_Directory_Traversal_File_Include

PREREQUISITES FOR THE ATTACK

Having an account with with Super Admin privileges in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

To upload a file to any location inside the Microsoft Azure Blob container, use a file-upload function available when, for example, creating a new reward, material, or news item, accessible to an account with Super Admin privileges:



When the file is added, the application sends the following request:

```
POST /web-panel/files?fileType=Image HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
Content-Type: multipart/form-data; boundary--------------------------
240813758136739653422361685593
Content-Length: 443

-----------------------------240813758136739653422361685593
Content-Disposition: form-data; name="file"; filename="test.png"
Content-Type: image/png

PNG
```

In response, the server returns the ID of the uploaded file:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

"77a5b3fe-f4a7-[…]"
```

To see the exact location of the uploaded file, send the following request:

```
GET /web-panel/files/77a[…]e-f4a7-[…] HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
```

Server response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

{
  "id" : "77[…]fe-f4a7-[…]",
  "fileUrl"    :    "https://[...]/abe9ad52-ebfa-[...]/image/7[…]b3fe-f4a7-[…]_test.png?sv=[…]-11-03&se=[…]-08-0[…]4%3A33%3A00Z&sr=b&sp=r&sig=x7upZ51K[…]1B8%2FM6qZl[...]",
  "fileType" : "Image",
  "fileName" : "[…].png",
  "contentType" : "image/png"
}
```

The uploaded „Image" file with the `.png` extension should, according to the application's design, be located in the `/{container}/image/` directory.

By using the `../` sequence it is possible to place the uploaded file in any directory within the container; for example, using the file name `/../../path-traversalSecuritum/aaaa` will create the file `aaaa.png` in the `{container}/path-traversalSecuritum/` directory.

Request sent:

```
POST /web-panel/files?fileType=Image HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
Content-Type: multipart/form-data; boundary=--------------------------
240813758136739653422361685593
Content-Length: 472

----------------------------240813758136739653422361685593
Content-Disposition: form-data; name="file"; filename="/../../path-traversalSecuritum/aaaa.png"
Content-Type: image/png

%PNG
```

Server response with the ID of the uploaded file:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

"0c93f739-0bf5-[…]
```

Preview of the exact location:

```
GET /web-panel/files/0c93f739-0bf5-[…] HTTP/2
Host: [HOSTNAME]
Authorization: Bearer […]
```

Server response:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel

{
  "id" : "0c93f739-0bf5-[…]",
  "fileUrl"            :            "https://[...].[DOMAIN]            /abe9ad52-ebfa-[...]/path-
traversalSecuritum/aaaa.png?sv=2023-11-03&se=2024-08-07T14%3A50%3A18Z&sr=b&sp=r&sig=[...]Ki8%3D",
  "fileType" : "Image",
  "fileName" : "/../../path-traversalSecuritum/aaaa.png",
  "contentType" : "image/png"
}
```

LOCATION

File-upload functionality:

- [HOSTNAME]/web-panel/files?fileType=Image

RECOMMENDATION

It is recommended to validate all the data received from the 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.

More information:

- https://owasp.org/www-community/attacks/Path_Traversal
- https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[FIXED] [LOW] SECURITUM-XXXXXX-007: Authentication via HTTP Basic Authentication

RETEST STATUS 29.01.2025

If the business requirements of the application assume public access to the Open API documentation, the vulnerability is considered remediated due to the intentional public availability of the resource: https://ca-Client_Nameapp-[REDACTED]azurecontainerapps.io/swagger/index.html.

SUMMARY

The tested application, when attempting to access Swagger at the address: [HOSTNAME]/swagger, uses HTTP Basic Authentication for authentication. This method introduces several security risks, including:

1) Authentication credentials must be sent with every HTTP request. As a result, even when a secure communication channel (e.g., HTTPS) is used, the attack surface significantly increases because the same credentials appear in multiple requests.
2) Once credentials are submitted using HTTP Basic Authentication, all web browsers retain this information until the browser window is closed. This means there is no straightforward way to "log out" of the application.
3) With HTTP Basic Authentication, there is no easy way to prevent browsers from saving the password locally. This poses a risk of credential theft by malware running on an infected machine.
4) If the web server hosting the application is not configured to use a centralized user management system, there is a risk of creating a shared username/password pair for multiple users. This is an unacceptable practice, as it makes it impossible to determine who accessed the system at a given time.
5) By default, a web server using HTTP Basic Authentication is not protected against brute-force attacks. Implementing such protection requires additional configuration (e.g., using tools like fail2ban), which may still be ineffective if shared credentials are used. Locking accounts temporarily in response to repeated login attempts may also result in a Denial of Service for other users.
6) The application lacks a password change feature. If a user's credentials are leaked, there is no quick way to change them.
7) It is possible to establish multiple simultaneous sessions using the same credentials. The system does not provide information about active sessions or any way to terminate them in the event of suspicious activity.

PREREQUISITES FOR THE ATTACK

Obtaining application access credentials by any means.

TECHNICAL DETAILS (PROOF OF CONCEPT)

When accessing the Swagger URL, a Basic Authentication login prompt is displayed:

During the allocated testing period, it was not possible to bypass security controls or gain access to Swagger.

LOCATION

[HOSTNAME]/swagger – authentication process.

RECOMMENDATION

It is recommended to stop using the HTTP Basic Authentication mechanism. To better secure access to Swagger, client certificates can be used. A detailed description of this method can be found at the link below:

- https://sekurak.pl/uwierzytelnianie-certyfikatem-klienckim-ssl-krotki-przewodnik/
- https://www.scriptjunkie.us/2013/11/adding-easy-ssl-client-authentication-to-any-webapp/

More information:

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

Informational issues

[NOT IMPLEMENTED] [INFO] SECURITUM-XXXXXX-008: Lack of Content-Security-Policy header

RETEST STATUS 29.01.2025

The recommendation has not been implemented. The Content Security Policy (CSP) is still missing from server responses.

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://sekurak.pl/wszystko-o-csp-2-0-content-security-policy-jako-uniwersalny-straznik-bezpieczenstwa-aplikacji-webowej/
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

LOCATION

[HOSTNAME]/

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 IMPLEMENTED] [INFO] SECURITUM-XXXXXX-009: Lack of Referrer-Policy header

RETEST STATUS 29.01.2025

The recommendation has not been implemented. The Referrer-Policy header is still missing from server responses.

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

[HOSTNAME]

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.

[NOT IMPLEMENTED] [INFO] SECURITUM-XXXXXX-010: Lack of Strict-Transport-Security (HSTS) header

RETEST STATUS 29.01.2025

The recommendation has not been implemented. The Strict-Transport-Security header is still missing from server responses.

SUMMARY

The HTTP header: `Strict-Transport-Security` (HSTS) was not identified in the application responses.

The introduction of HSTS forces the browser to use an encrypted HTTPS connection in all references to the application domain. Even manually entering the "HTTP" protocol name in the address bar will not send unencrypted packets.

The implementation of this header is treated as a generally good practice for hardening web application security.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
- https://sekurak.pl/hsts-czyli-http-strict-transport-security/

LOCATION

[HOSTNAME]/

RECOMMENDATION

The server's HTTP responses should contain a header:

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

Alternatively, it is possible to define the HSTS header for all subdomains:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

In addition, it is possible to use the so-called preload list, which by default is saved in the sources of popular web browsers. The result is that the user's browser, which connects to the application for the first time, will immediately enforce the use of an encrypted, secure communication channel. The preload value is set as follows:

```
Strict-Transport-Security: max-age=31536000; preload
```

More information:

- https://hstspreload.org/
- https://www.chromium.org/hsts
- https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

[NOT IMPLEMENTED] [INFO] SECURITUM-XXXXXX-011: Lack of X-Content-Type-Options header

RETEST STATUS 29.01.2025

The recommendation has not been implemented. The X-Content-Type-Options header is still missing from server responses.

SUMMARY

The `X-Content-Type-Options` header was not identified in the responses of the application.

This header protects from attacks based on the so-called MIME-sniffing, i.e. guessing the MIME type of response by web browser based on the content of the received response, instead of a `Content-Type` header value. This may lead to the browser being forced to load the resource as HTML, even if its type is e.g. `application/json`. As a result, an XSS attack may be performed.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

LOCATION

[HOSTNAME]/

RECOMMENDATION

The following header should be added in all server responses:

```
X-Content-Type-Options: nosniff
```

[NOT IMPLEMENTED] [INFO] SECURITUM-XXXXXX-012: No invalidation of the session after logout

RETEST STATUS 29.01.2025

The recommendation has not been implemented. The logout operation does not invalidate the JWT token.

SUMMARY

During the audit, it was found that the session is not invalidated when the “Log out” button is pressed, the token used for API interaction is not invalidated. An attacker who successfully acquires the “access_token” (“Bearer”) identifier will be able to use it to interact with the API.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to confirm the vulnerability, the following steps need to be performed:

1. A user logs into the application.
2. The user presses the “Log out” button.
3. An attacker in any way acquires the “access_token” (“Bearer”) token.
4. The attacker sends a request with the “inactive” token:

```
GET /web-panel/companies/names HTTP/2
Host: [HOSTNAME]
Authorization: Bearer eyJhbGciOiJSUz[…REDACTED…]DdjlS0XOSbXF
```

5. In response, the application returns the data presented below and thus it is confirmed that the token is still active:

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
[…]
Server: Kestrel
Access-Control-Allow-Origin: *

{"names" : [ […] ] }
```

LOCATION

Session management.

RECOMMENDATION

It is recommended to invalidate user's session immediately after the “Log out” button is pressed.