securitum

Security report

SUBJECT

„[APPNAME]” Web Application

DATE

10.07.2023 – 19.07.2023

RETEST DATE

N/A

LOCATION

Poland

AUTHOR

Mateusz Lewczak

VERSION

1.0

Executive summary

This document is a summary of work conducted by the Securitum. The subject of the test was the „[APPNAME]" web application available at https:// [APPNAME]/.

Tests were conducted using the following roles:

- user with full access to the course,
- user with access to demo version,
- superadmin,
- unauthenticated user (visitor of the website).

The most severe vulnerabilities identified during the assessment were:

- [MEDIUM] SECURITUM-234069-001: Authorization – ability to change another user's avatar,
- [MEDIUM] SECURITUM-234069-002: Authorization – ability to send messages to the chat room without being a participant.

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, ffuf, nmap), was used during the assessment. The security audit was conducted using a best effort approach, prioritizing the critical aspects of the application within the given time constraints.

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

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.

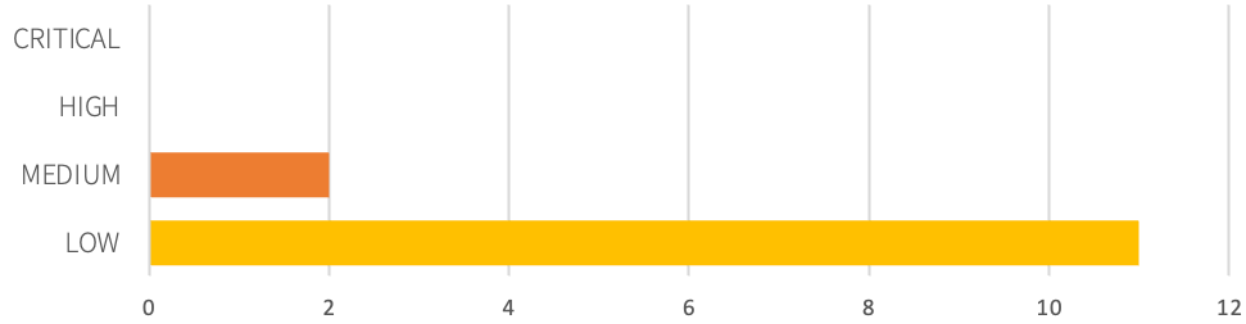
- HIGH – exploitation of the vulnerability makes it possible to access sensitive data (similar to the 'CRITICAL' level), however the prerequisites for the attack (e.g. possession of a user account in an

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, 9 INFO issues are reported.

Contents

Change history

Document date	Version	Change description
19.07.2023	1.0	Create a document.

Vulnerabilities in the web application

[MEDIUM] SECURITUM-234069-001: Authorization – ability to change another user's avatar

SUMMARY

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

By exploiting this vulnerability, it was possible to change another user's avatars.

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

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to change another user's avatar, the following steps have to be performed:

1. Log in to the application using any account.
2. Open menu in the right upper corner,
3. Go to „KONTO" > „Profil publiczny",
4. Set up a tool to intercept HTTP requests (e.g. Burp Suite),
5. Click „ZMIEŃ AVATAR" and select any image and capture HTTP request:

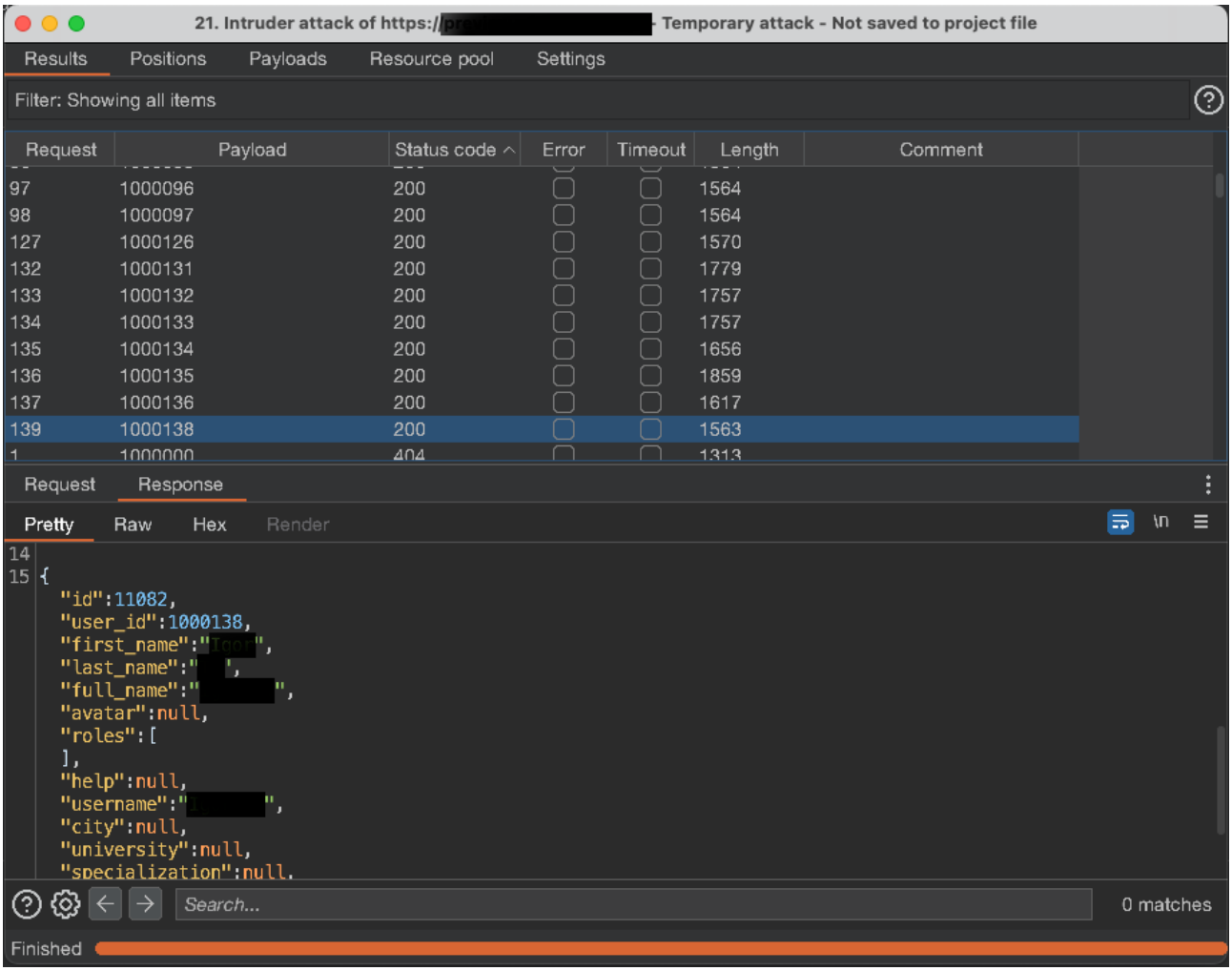
```
POST /papi/v2/users/current/avatar HTTP/2
Host: [APPNAME]
Cookie: remember_web_preview=[...]; XSRF-TOKEN=[...]; laravel_session=[...]
Content-Length: 342
X-Xsrf-Token: [...]
X-Csrf-Token: [...]
User-Agent: Mozilla/5.0
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryOvH74CgGdY4rbLkw
Accept: application/json
X-Client_Name-Location: https://[APPNAME]/app/myself/profile
X-Requested-With: XMLHttpRequest
Origin: https://[APPNAME]
Referer: https://[APPNAME]/app/myself/profile

------WebKitFormBoundaryOvH74CgGdY4rbLkw
Content-Disposition: form-data; name="file"; filename="smallest.png"
Content-Type: image/png

[PNG FILE CONTENT]
------WebKitFormBoundaryOvH74CgGdY4rbLkw--
```

6. Now change the value from `current` to the value of the victim identifier. Due to the use of numeric identifiers and the fact that each user profile is public to other authenticated users (with any

privileges), it is possible to simply enumerate the victim profile using the following request (the place of enumeration is marked in yellow):

```
GET /papi/v2/users/1000131/profile HTTP/2
Host: [APPNAME]
Cookie: XSRF-TOKEN=[...]; laravel_session=[...];
User-Agent: Mozilla/5.0
Accept: application/json
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
X-Csrf-Token: [...]
X-Client_Name-Location: https://[APPNAME]/app/users/1000131
X-Requested-With: XMLHttpRequest
X-Xsrf-Token: [...]
Referer: https://[APPNAME]/app/users/1000131
```

Below is the result of enumeration using the Intruder tool in Burp Suite Professional:



LOCATION

POST /papi/v2/users/{ID}/avatar

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

[MEDIUM] SECURITUM-234069-002: Authorization – ability to send messages to the chat room without being a participant

SUMMARY

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

By exploiting this vulnerability, it was possible to send messages to the chat room without being a participant.

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

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

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

1. Log in to the application using any account.
2. Open a chat view, with any user.
3. Set up a tool to intercept WebSocket messages (e.g. Burp Suite),
4. Send any message to another user and capture WebSocket message (notice room id):


```
42["sendMessage",{"room":{"id":3489,"channel":"private-1000135-
1000131","unread_count":0,"last_message_time":1689604253432,"type":"private","profiles":[11080,11
076],"pagination":{"current":1689246953141,"next":null,"previous":null,"has_more":false}},"messag
e":{"content":"<p>asd</p>"},"users":[{"id":11080,"user_id":1000135,"first_name":"Anon","last_name
":"1.","full_name":"Anon                    1000135","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/E0hTbCuu5H02vsIXCHUelRVRgs5S88C6.png","roles":["superadmin","moder
ator","admin","moderator"],"help":null,"username":"Anon
1000135","city":null,"university":null,"specialization":null,"interests":null,"about":null,"learn
ing_location":null,"deleted_at":null,"chat_rooms":3489},{"id":11076,"user_id":1000131,"first_name
":"Anon","last_name":"1.","full_name":"Anon      1.","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png","roles":[],"help":"asd","use
rname":"Anon
1.","city":"asd","university":"asd","specialization":"asd","interests":"aasdarrrrr","about":"aasd
arrrrr","learning_location":"asda","deleted_at":null,"chat_rooms":3487}]}]
```


5. Change the selected room identifier (it is numeric) to another arbitrary one and forward the message.


```
42["sendMessage",{"room":{"id":3490,"channel":"private-1000135-
1000131","unread_count":0,"last_message_time":1689604253432,"type":"private","profiles":[11080,11
076],"pagination":{"current":1689246953141,"next":null,"previous":null,"has_more":false}},"messag
e":{"content":"<p>asd</p>"},"users":[{"id":11080,"user_id":1000135,"first_name":"Anon","last_name
":"1.","full_name":"Anon                    1000135","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/E0hTbCuu5H02vsIXCHUelRVRgs5S88C6.png","roles":["superadmin","moder
ator","admin","moderator"],"help":null,"username":"Anon
1000135","city":null,"university":null,"specialization":null,"interests":null,"about":null,"learn
ing_location":null,"deleted_at":null,"chat_rooms":3489},{"id":11076,"user_id":1000131,"first_name
```

```
":"Anon","last_name":"1.","full_name":"Anon      1.","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png","roles":[],"help":"asd","use
rname":"Anon
1.","city":"asd","university":"asd","specialization":"asd","interests":"aasdarrrrr","about":"aasd
arrrrr","learning_location":"asda","deleted_at":null,"chat_rooms":3487}]}]
```


6. As a result of the above action, there is a message from a user who is not part of the chat with ID 3490:

In the screenshot above, it can be seen that the auditor is logged into an account with a gray avatar, and the other participant is user `Anon 20001`. However, in the chat, a message from a user with a blue avatar can be seen.

LOCATION

Chat mechanism.

RECOMMENDATION

It is recommended to 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

[LOW] SECURITUM-234069-003: Authorization – demo user can modify his or her public profile

SUMMARY

The tested application does not implement proper authorization of access to data; thus any demo user may modify his or her profile. Normally, a demo user does not have the ability to modify his public profile.

By exploiting this vulnerability, it was possible to modify public profile as a demo user. Normally, the user has no such function.

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

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to modify demo user's public profile, the following steps have to be performed:

1. Log in to the application using demo account.
2. Send the following HTTP request, but change the values of Cookies and CSRF tokens:

```
PUT /papi/v2/users/current/profile HTTP/2
Host: [APPNAME]
Cookie: XSRF-TOKEN=[DEMO USER XSRF TOKEN]; laravel_session=[DEMO USER SESSION TOKEN];
Content-Length: 393
X-Xsrf-Token: [DEMO USER XSRF TOKEN]
X-Csrf-Token: [DEMO USER CSRF TOKEN]
Content-Type: application/json;charset=UTF-8

{"help":"help","specialization":"specialization","university":"university","city":"city","learnin
g_location":"asda","interests":"interests","about":"aasdarrrrr","first_name":"Demo","last_name":"
User","full_name":"Demo                          User","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png","roles":[],"username":"Demo"
,"deleted_at":null}
```

3. The application will return success information in response:

```
HTTP/2 200 OK
Content-Type: application/json
[...]

[]
```

LOCATION

https://[APPNAME]/papi/v2/users/current/profile

RECOMMENDATION

It is recommended to 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 or has privileges to modify.

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

[LOW] SECURITUM-234069-004: Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript code

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 [APPNAME] domain. This behaviour can be used, among other things, to extract and steal any data from the application.

Note: There are many places in the admin panel where JavaScript code can be injected, especially where there is a possibility to edit content in HTML. Due to the nature of the audit (best effort approach), details of two examples are given.

Below is a list of places where one can inject JavaScript code:

- Page editing,
- Lesson editing,
- Dashboard news (executes in dashboard),
- Closed questions editing (executes in the question database),
- Opened questions editing (executes in revision module),
- User's name (executes in "Pomoc" tab).

Due to the required administrator privileges and the fact that the session cookie has the `HttpOnly` flag, this vulnerability has been classified as LOW.

More information:

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

PREREQUISITES FOR THE ATTACK

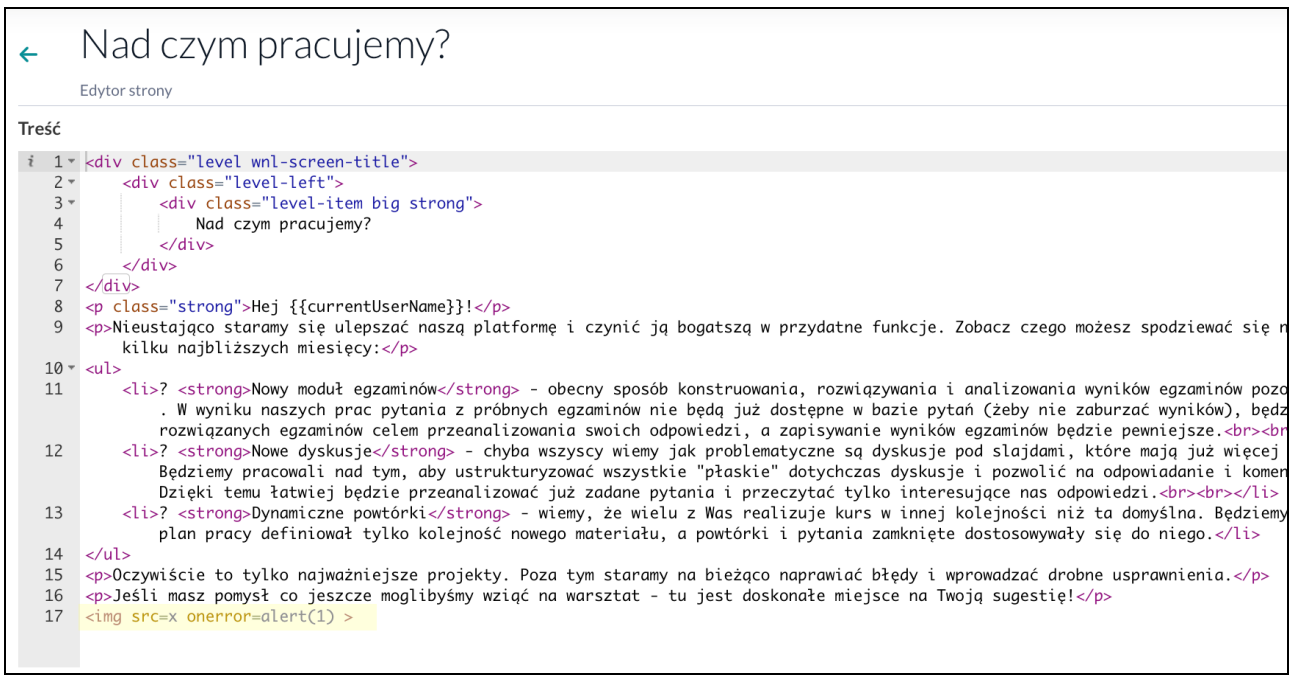
Access to admin panel.

TECHNICAL DETAILS (PROOF OF CONCEPT)

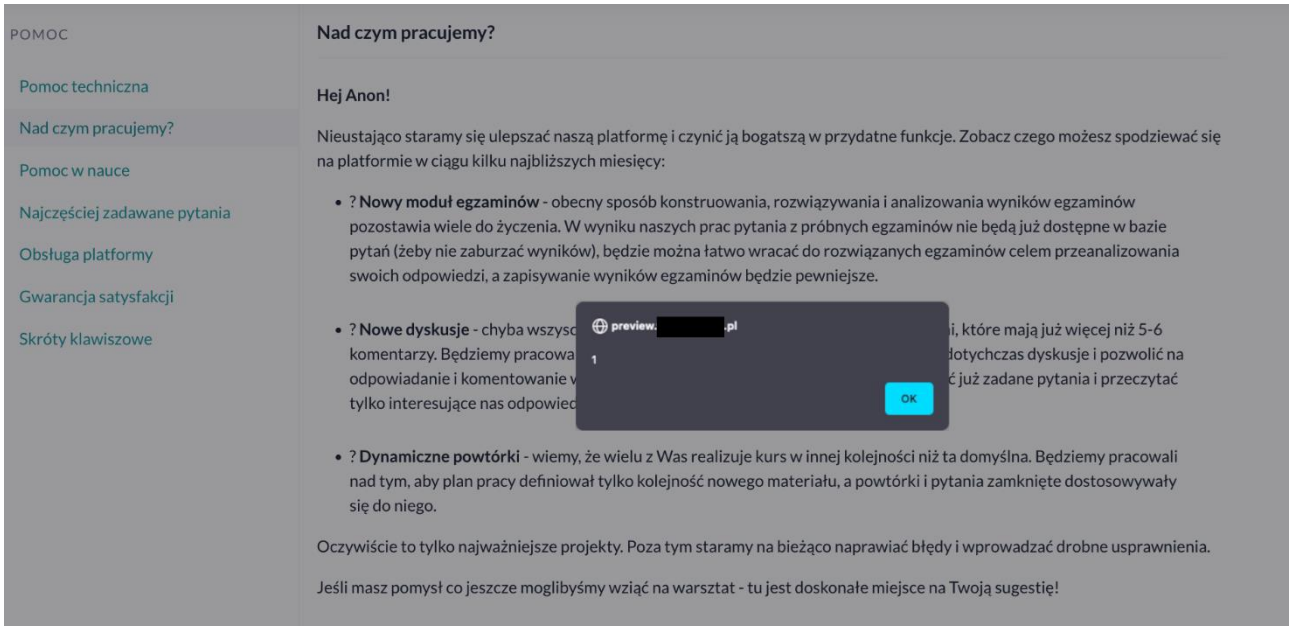
Case #1 – Page editing

To successfully exploit the vulnerability, these steps need to be followed:

1. Log into the application using an account with admin privileges.
2. Go to Admin Panel > "Lista stron" and then pick any page and click "Edytuj".
3. At the end of the "Treść" textarea paste JavaScript code:

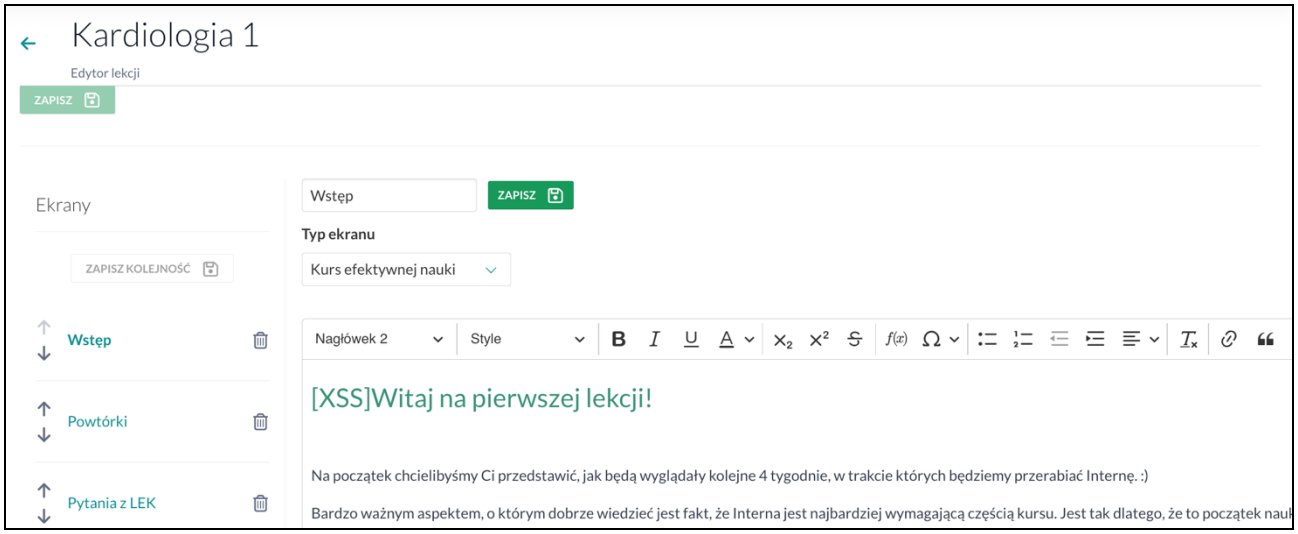


4. Log into the application using an account with normal privileges.
5. Go to “Pomoc” and select targeted page:



Case #2 – Lesson Editing

1. To successfully exploit the vulnerability, these steps need to be followed:
2. Log into the application using an account with admin privileges.
3. Go to Admin Panel > “Edycja lekcji” and then pick any topic and then any screen.
4. Set up a tool to intercept HTTP requests (e.g. Burp Suite),
5. Add text `[XSS]` at the beginning of the content.



6. Save changes and capture HTTP request:

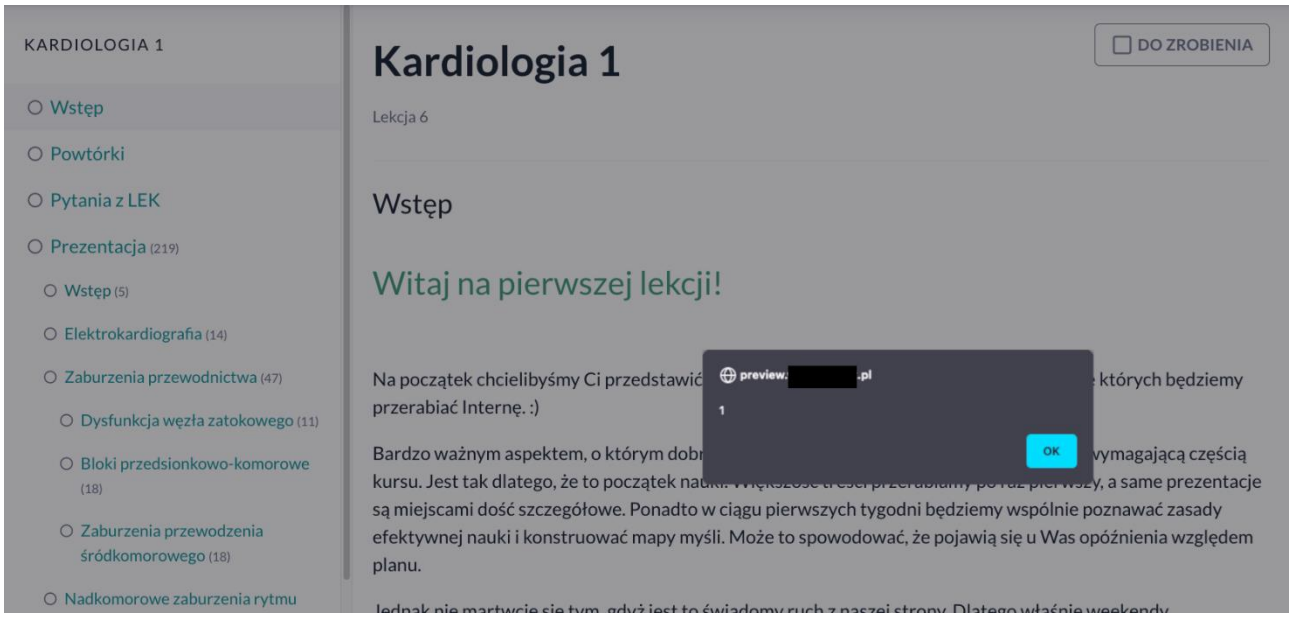
```
PUT /papi/v2/screens/18?include=screenables HTTP/2
Host: [APPNAME]
Cookie: remember_web_preview=[...]; XSRF-TOKEN=[...]; laravel_session=[...]
Content-Length: 6911
X-Xsrf-Token: [...]
X-Csrf-Token: [...]
User-Agent: Mozilla/5.0
Content-Type: application/json;charset=UTF-8
Accept: application/json
X-Client_Name-Location: https://[APPNAME]/admin/app/lessons
X-Requested-With: XMLHttpRequest
Origin: https://[APPNAME]
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

{"content":"<h2><span style=\"color:rgb(56,148,118);\">[XSS]Witaj na pierwszej
lekcji!</span></h2><p> </p>[LONG_TEXT]","meta":"{}","name":"Wstęp","type":"effective_learnin
g","screenables":[],"id":18,"order_number":0,"tags":[],"lessons":1,"discussion_id":null,"is_discu
ssable":0}
```

7. Change the placeholder to XSS payload and forward the request:

```
PUT /papi/v2/screens/18?include=screenables HTTP/2
Host: [APPNAME]
Cookie: remember_web_preview=[...]; XSRF-TOKEN=[...]; laravel_session=[...]
Content-Length: 6911
X-Xsrf-Token: [...]
X-Csrf-Token: [...]
User-Agent: Mozilla/5.0
Content-Type: application/json;charset=UTF-8
Accept: application/json
X-Client_Name-Location: https://[APPNAME]/admin/app/lessons
X-Requested-With: XMLHttpRequest
Origin: https://[APPNAME]
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

```
{"content":"<h2><span style=\"color:rgb(56,148,118);\"><img src=x onerror=alert(1)/>Witaj na pierwszej
lekcji!</span></h2><p> </p>[LONG_TEXT]","meta":"{}","name":"Wstęp","type":"effective_learnin
g","screenables":[],"id":18,"order_number":0,"tags":[],"lessons":1,"discussion_id":null,"is_discu
ssable":0}
```


8. Log into the application using an account with normal privileges.
9. Go to “Kurs” and select targeted lesson and screen:



LOCATION

https://[APPNAME]/admin/app/

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

[LOW] SECURITUM-234069-005: Weak password policy

SUMMARY

During the tests, it was observed that the application has weak password policy implemented. Weak policy allows users to set simple passwords that can then be cracked by an attacker.

More information:

- https://wiki.owasp.org/index.php/Testing_for_Weak_password_policy_(OTG-AUTHN-007)
- https://cwe.mitre.org/data/definitions/521.html

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During the tests, it was possible to set a password consisting only of numbers:

```
PUT /papi/v2/users/current/password HTTP/2
Host: [APPNAME]
Cookie: _ga=[...]; _ga_EG07RW05XT=[...]; XSRF-TOKEN=[...]; laravel_session=[...]; _gid=[...]
User-Agent: Mozilla/5.0
Accept: application/json
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
X-Csrf-Token: [...]
X-Client_Name-Location: https://[APPNAME]/app/myself/password
X-Requested-With: XMLHttpRequest
Content-Type: application/json;charset=utf-8
X-Xsrf-Token: [...]
Content-Length: 124
Origin: https://[APPNAME]

{"old_password":"[...]","new_password":"12345678","new_password_confirmation":"12345678"}
```

The application returns a success message in response:

```
HTTP/2 200 OK
Cache-Control: max-age=0, no-store, private
Content-Type: application/json
Date: Mon, 17 Jul 2023 12:31:53 GMT
Server: nginx
Set-Cookie: [...]
Vary: Accept-Encoding
X-Client_Name-App-Public-Backend-Version: 15
X-Client_Name-User-Course-Access-Status: active
X-Powered-By: PHP/8.1.19
X-Response-Time-Ms: 249.587
X-Served-By: platform-app-5d475d48b9-88mqk

[]
```

LOCATION

PUT /papi/v2/users/current/password

RECOMMENDATION

It is recommended to implement the requirements regarding password policy, in particular:

a) Enforcing a minimum password length of at least 12 characters and a maximum length of up to 128 characters (length limitation should be introduced due to potential DoS attacks in the absence of it);
b) Checking if the password is not present in at least 10,000 of the most popular passwords from database leaks and other sources, as well as in publicly available password dictionaries (most commonly used for brute-force attacks);
c) Lack of requirements regarding the complexity of the password and thus no restrictions on the types of characters;
d) Enforcing the need to change the password in case of suspected compromise;
e) Requirement to provide the current password if it is being changed;
f) Lack of an option to remind password based on known elements (it is forbidden to use questions like "What was the name of your first car?");
g) Detection of mass login attempts to one account with different passwords or to many accounts with one password provided; after a maximum of 5 unsuccessful login attempts, additional verification should be entered (e.g. using CAPTCHA codes); alternatively, the account can be blocked temporarily, although with this solution it should be taken into consideration that an attacker could intentionally block accounts;
h) Implementation of a mechanism (if it does not exist) of remote blocking of a given user account (blocking should also automatically log out the user from all systems);
i) Implementation of an application-based two-factor authentication (2FA), e.g. Google Authenticator (using SMS is absolutely not recommended).

It should be noted that some systems still force the configuration of password complexity or expiration. Therefore, as a transitional solution (not considered as completely secure) the following can be set temporarily (until the above policy is fully implemented):

a) Implementation of the password complexity requirements to contain a minimum of 1 special character, 1 digit, 1 lowercase letter and 1 uppercase letter;
b) Enforcing a periodic password change every 1 year.

It should be taken into account, that the implementation of only some points from the recommendations will also be considered not completely safe, therefore it is recommended to implement them all.

Access to sensitive functionalities and systems should always force reauthentication.

It is also worth considering implementing functionality that will verify the strength of the password – this helps to limit the risk that users will create simple passwords despite the restrictions.

More information:

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

[LOW] SECURITUM-234069-006: Redundant information disclosure about the application environment in HTTP response header

SUMMARY

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

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of the HTTP request sent to the application:

```
GET / HTTP/2
Host: [APPNAME]
```

In response, the application returns:

```
HTTP/2 302 Found
[...]
X-Powered-By: PHP/8.1.19
X-Response-Time-Ms: 20.304
X-Served-By: platform-app-5d475d48b9-h4b8s
```

LOCATION

https://[APPNAME]/

RECOMMENDATION

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

[LOW] SECURITUM-234069-007: Redundant information disclosure in PDF metadata in generated files

SUMMARY

The metadata of PDF files generated in the tested application contains 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.

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

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

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

1. Log into the application.
2. Open menu in the right upper corner,
3. Go to „KONTO” > „Twoje zamówienia” > „Faktury”,
4. Download invoice.
5. Use `exiftool` to check metadata:

```
$ exiftool FV_20777.pdf
ExifTool Version Number         : 12.42
File Name                       : FV_20777.pdf
Directory                       : .
File Size                       : 39 kB
File Modification Date/Time     : 2023:07:18 12:24:33+02:00
File Access Date/Time           : 2023:07:18 12:24:33+02:00
File Inode Change Date/Time     : 2023:07:18 12:24:33+02:00
File Permissions                : -rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.7
Linearized                      : No
Page Count                      : 1
Producer                        : dompdf 1.2.0 + CPDF
Create Date                     : 2023:07:17 14:08:40+00:00
Modify Date                     : 2023:07:17 14:08:40+00:00
Title                           : Faktura VAT
```

LOCATION

PDF files generators, e.g.:

- Proforma Invoice,

- VAT invoice.

RECOMMENDATION

It is recommended to remove from PDF files redundant metadata that reveals information about the technologies in use.

[LOW] SECURITUM-234069-008: Redundant or default file publicly available

SUMMARY

During the audit, it was discovered that a `.htaccess` configuration file was available. This file could expose information about the system. This behaviour can help an attacker to better profile the application environment, which can then be used to carry out further attacks.

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

To open an `.htaccess` file, just open the URL below in a browser:

```
https://[APPNAME]/.htaccess
```

As a result of this operation, a file will be downloaded:

```
<IfModule mod_rewrite.c>
    <IfModule mod_negotiation.c>
        Options -MultiViews
    </IfModule>

    RewriteEngine On

    # Redirect Trailing Slashes If Not A Folder...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule ^(.*)/$ /$1 [L,R=301]

    # Handle Front Controller...
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]

    # Handle Authorization Header
    RewriteCond %{HTTP:Authorization} .
    RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
</IfModule>
```

LOCATION

https://[APPNAME]/.htaccess

RECOMMENDATION

It should be verified if the file needs to be publicly accessible. If not, it should be removed.

[LOW] SECURITUM-234069-009: Lack of security attributes for sensitive cookies

SUMMARY

During the audit it was observed that the application did not set security attributes for sensitive cookies. These attributes are:

- `SameSite` – could prevent browser from sending a cookie between pages with different sites. Implementing this attribute could be helpful, among others, in preventing CSRF attacks;
- `Secure` – informs the browser to send the cookie only using an encrypted communication channel (HTTPS). If this attribute is not set and an attacker intercepts unencrypted communication, he or she can potentially access the cookie value, which can result in account takeover.

Below is the list of cookies with no security attributes set:

- XSRF-TOKEN,
- laravel_session.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#cookies
- https://wiki.owasp.org/index.php/Testing_for_cookies_attributes_(OTG-SESS-002)
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
- https://cwe.mitre.org/data/definitions/1004.html
- https://cwe.mitre.org/data/definitions/614.html

PREREQUISITES FOR THE ATTACK

Presence of another, exploitable vulnerability (e.g. XSS, CSRF, MitM).

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example request sent to the application:

```
GET / HTTP/2
Host: [APPNAME]
```

Response from the server with `Set-Cookie` header and cookies without security attributes:

```
HTTP/2 302 Found
Cache-Control: no-cache, private
Content-Type: text/html; charset=UTF-8
Date: Tue, 18 Jul 2023 10:08:49 GMT
Location: https://[APPNAME]/login
Server: nginx
Set-Cookie: XSRF-TOKEN=[...]; expires=Wed, 19-Jul-2023 10:08:49 GMT; Max-Age=86400; path=/; domain=.[APPNAME]
Set-Cookie: laravel_session=[...]; expires=Wed, 19-Jul-2023 10:08:49 GMT; Max-Age=86400; path=/; domain=.[APPNAME]; httponly
X-Powered-By: PHP/8.1.19
X-Response-Time-Ms: 20.304
X-Served-By: platform-app-5d475d48b9-h4b8s
```

```
[...]
```

Although the XSRF-TOKEN cookie does not have the SameSite flag set, it is impossible to perform a Cross-Site Request Forgery attack due to the need to add this cookie and an additional CSRF token in the request header.

LOCATION

Cookie management.

RECOMMENDATION

It is recommended that the application sets the `httpOnly`, `SameSite` and `Secure` attributes for sensitive cookies, where `SameSite` attribute should be set to one of the following values:

- `Strict` – the browser will not send the cookie in cross-site requests,
- `Lax` – the browser will send the cookie in cross-site requests if and only if it is sent using safe HTTP method (GET, HEAD, OPTIONS, TRACE) and it is top-level navigation (i.e. the address bar will show the change of the domain); in other cases of cross-site requests, the cookie will not be sent. In modern browsers, this is the default value if `SameSite` attribute has not been specified explicitly.

E.g.:

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

More information:

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

[LOW] SECURITUM-234069-010: Outdated software

SUMMARY

It was observed that many 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://security.snyk.io/package/npm/jquery/3.2.1
- https://security.snyk.io/package/composer/dompdf%2Fdompdf
- 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://[APPNAME]/build/assets/design-system.umd-650715c7.js	jQuery 3.2.1
PDF generating	dompdf 1.2.0

LOCATION

Listed in the technical details.

RECOMMENDATION

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

More information:

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

[LOW] SECURITUM-234069-011: Redundant data in admin panel

SUMMARY

During the audit, it was discovered that the admin panel contains redundant access data for the Przelewy24 service. This data can be used by an attacker to cause financial damage.

PREREQUISITES FOR THE ATTACK

Access to the admin panel.

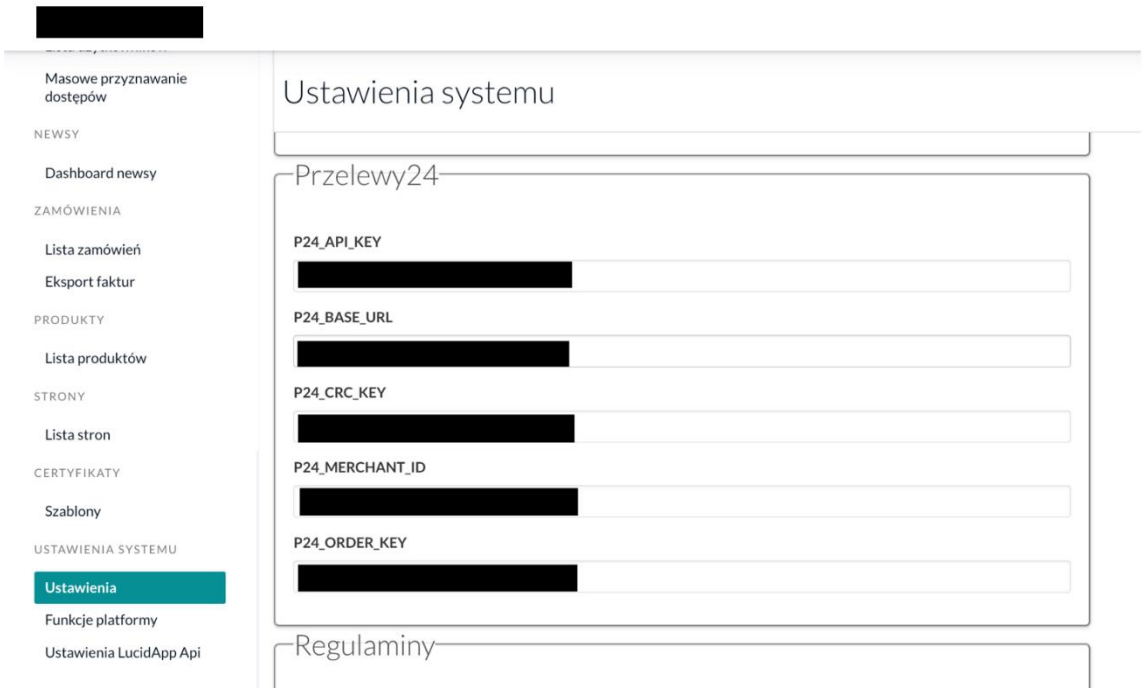
TECHNICAL DETAILS (PROOF OF CONCEPT)

To find the mentioned data, the steps below need to be followed:

1. Go to the following URL:

```
https://[APPNAME]/admin/app/settings
```

2. Scroll down to section “Przelewy24”:



Here one will find access data and data for generating transactions for the Przelewy24 service.

LOCATION

https://[APPNAME]/admin/app/settings

RECOMMENDATION

It is recommended that sensitive data (such as credentials) are not returned in the server response. It should only be possible to overwrite this data, without being able to view it. Changing the input type in the HTML code to "password" is not a solution to the problem as in such case confidential data can still be viewed in the page source code.

[LOW] SECURITUM-234069-012: Support for outdated TLS cipher suites

SUMMARY

The tested application supports weak TLS cipher suites, 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://cheatsheetseries.owasp.org/cheatsheets/TLS_Cipher_String_Cheat_Sheet.html
- 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 weak TLS cipher suites:

```
TLSv1.2
ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES256-SHA
AES128-GCM-SHA256
AES256-GCM-SHA384
AES128-SHA
AES256-SHA
ECDHE-RSA-DES-CBC3-SHA
DES-CBC3-SHA
```

LOCATION

https://[APPNAME]/ (443/tcp)

RECOMMENDATION

It is recommended to disable support for the TLS cipher suites 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_Protection_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/TLS_Cipher_String_Cheat_Sheet.html

[LOW] SECURITUM-234069-013: HTML injection

SUMMARY

It was found, that in the process of sending messages to chat root, it is possible to inject HTML tags into the message. The HTML code is executed in the browser of a recipient. An attacker can use this method and the fact of the SECURITUM-234069-002 vulnerability to perform a phishing attack.

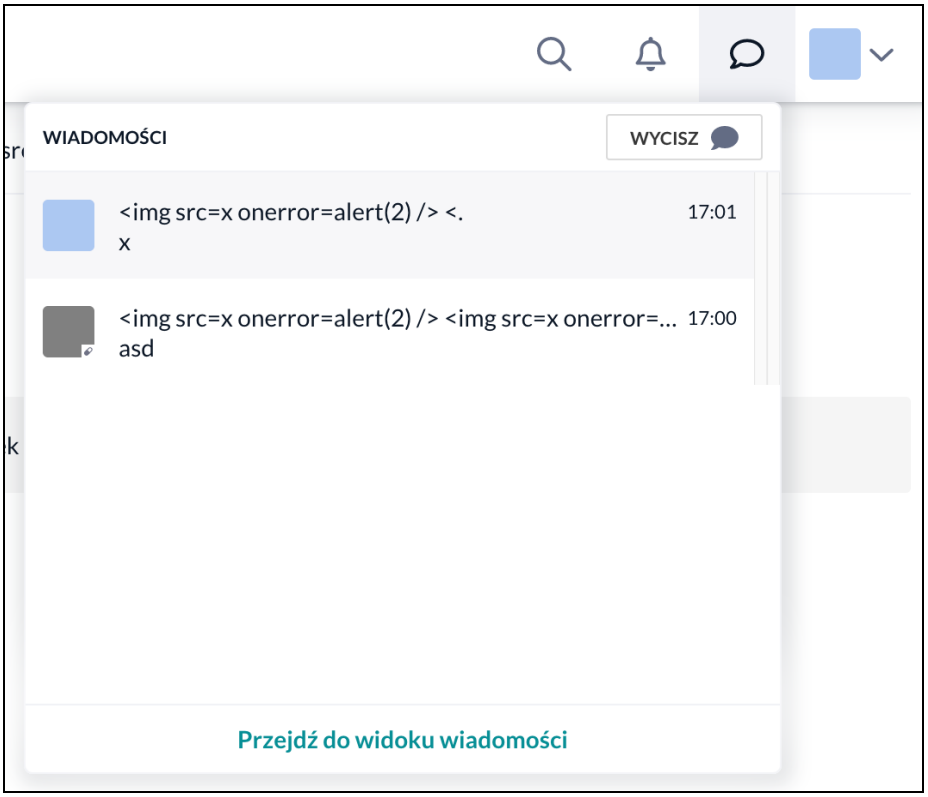
PREREQUISITES FOR THE ATTACK

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

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

1. Log into the application.
2. Open any chat room in the right upper corner:



3. Set up a tool to intercept WebSocket messages (e.g. Burp Suite),
4. Send any message to another user and capture WebSocket message (notice message):

```
42["sendMessage",{"room":{"id":3487,"channel":"private-
1000131","unread_count":0,"last_message_time":1689693249378,"type":"private","profiles":[11076],"
pagination":{"current":1689692422714,"next":null,"previous":null,"has_more":false}},"message":{"c
ontent":"<p>Inject   Here</p>"},"users":[{"id":11076,"user_id":1000131,"first_name":"<img   src=x
onerror=alert(2)    />","last_name":"<.","full_name":"<img    src=x    onerror=alert(2)    />
<.","avatar":"https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png","roles":[],"help":"asd","use
rname":"<img                    src=x                    onerror=alert(2)                    />
```

```
<.","city":"asd","university":"asd","specialization":"asd","interests":"aasdarrrrr","about":"aasd
arrrrr","learning_location":"asda","deleted_at":null,"chat_rooms":3489}]}]
```

5. Change placeholder as follows:

```
42["sendMessage",{"room":{"id":3487,"channel":"private-
1000131","unread_count":0,"last_message_time":1689693249378,"type":"private","profiles":[11076],"
pagination":{"current":1689692422714,"next":null,"previous":null,"has_more":false}},"message":{"c
ontent":"<img src='https://bucket-name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png'
/>"},"users":[{"id":11076,"user_id":1000131,"first_name":"<img src=x onerror=alert(2)
/>","last_name":"<.","full_name":"<img src=x onerror=alert(2) /> <.","avatar":"https://bucket-
name.s3.eu-central-
1.amazonaws.com/public/avatars/VKVoHRoTJn1OyhaR1iPCn6Eq1oBmYUua.png","roles":[],"help":"asd","use
rname":"<img src=x onerror=alert(2) />
<.","city":"asd","university":"asd","specialization":"asd","interests":"aasdarrrrr","about":"aasd
arrrrr","learning_location":"asda","deleted_at":null,"chat_rooms":3489}]}]
```

As a result of the above action, a picture will be shown in the time room:

Alternatively, the attacker can use this to send a link to the user. Sha or he can use the payload below, which will redirect the victim to his site after clicking the link:

```
<a href="http://attacker.site">Click me!</a>
```

And in the chat window there will be a link visible to the other participant:

LOCATION

Chat mechanism.

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 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

Informational issues

[INFO] SECURITUM-234069-014: Lack of Cache Control headers

SUMMARY

During the testing, the responses did not contain implemented Cache Control headers, which are responsible for preventing the saving of data related to the application permanently in the temporary memory of the browser. Such HTTP headers should include:

- Cache-Control,
- Pragma,
- Last-Modified,
- Expires.

`Cache-Control` is a header containing specific directives for the caching engine that are supposed to instruct the browser: which elements, and whether they should be stored in the temporary storage. Although some directives may also be used in a HTTP request, it is recommended to focus mainly on HTTP responses.

`Pragma` in case of using the "no-cache" directive means the same as "Cache-Control: no-cache" – that is, forces the browser to send a query to the server and download the current version before downloading the version that is saved in the browsers cache.

`Last-Modified` contains the exact date and time at which the network resource was last modified – the value should always correspond to the current time.

`Expires` contains the date, time or value that determines when the server response is no longer valid. Frequently used values in the form of a past date or "0" mean that the resource is no longer valid.

For more information on implementing caching headers, please visit:

- https://wiki.owasp.org/index.php/Testing_for_Browser_cache_weakness_(OTG-AUTHN-006)
- https://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html
- https://cwe.mitre.org/data/definitions/524.html
- https://cwe.mitre.org/data/definitions/525.html

LOCATION

https://[APPNAME]/

RECOMMENDATION

It is recommended to implement Cache Control headers in the server response, according to ASVS standard:

```
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Last-Modified: {current_time} GMT
Expires: Tue, 07 Jul 2001 07:00:00 GMT
```

More information:

- https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

[INFO] SECURITUM-234069-015: Lack of Content-Security-Policy header

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 the following CSP policy is defined:

```
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://[APPNAME]/

RECOMMENDATION

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

[INFO] SECURITUM-234069-016: 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://[APPNAME]/

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.

[INFO] SECURITUM-234069-017: Lack of Strict-Transport-Security (HSTS) header

SUMMARY

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

The introduction of HSTS forces a 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

LOCATION

https://[APPNAME]/

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
```

More information:

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

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. To use this option, the appropriate parameter to the HSTS header needs to be added and a submission needs to be applied via a dedicated form https://hstspreload.org/.

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security#preloading_strict_transport_security
- https://www.chromium.org/hsts

It is worth mentioning that preload lists are publicly available, so they should only be used when the domain can be publicly known.

[INFO] SECURITUM-234069-018: Lack of X-Content-Type-Options header

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

https://[APPNAME]/

RECOMMENDATION

The following header should be added in all server responses:

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

[INFO] SECURITUM-234069-019: 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://[APPNAME]/papi/

RECOMMENDATION

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

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

[INFO] SECURITUM-234069-020: No protection against Clickjacking attack – lack of X-Frame-Options header

SUMMARY

The analysis showed that the application has no protection against Clickjacking attack. The attack consists in placing a WWW page in a floating frame (iframe) by an attacker, which by covering up certain elements and functionality of the page can cause the victim of the attack to perform an unauthorized operation.

More information:

- https://owasp.org/www-community/attacks/Clickjacking
- https://owasp.org/www-community/attacks/Cross_Frame_Scripting

LOCATION

https://[APPNAME]/

RECOMMENDATION

It is recommended that the application sets the "X-Frame-Options" header for each page, choosing one of the following options:

a) Complete blocking of the page in the frame:

```
X-Frame-Options: DENY
```

b) Possibility to place a page in the frame by the target domain only:

```
X-Frame-Options: SAMEORIGIN
```

In addition, it is worth considering adding (as additional protection) JavaScript script that will verify that the page was not embedded in a frame – however, it should be ensured that the script does not create new vulnerabilities, such as Open Redirect or Cross-Site Scripting.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

[INFO] SECURITUM-234069-021: Numeric resource identifiers

SUMMARY

It was noticed that the application uses numeric resource identifiers (e.g., a user profile with ID 1000131). This approach in itself does not currently constitute a security vulnerability (the application additionally verifies the session), but in the event of a vulnerability of unauthorized access to resources, the predictability of the identifiers greatly facilitates an attack.

PREREQUISITES FOR THE ATTACK

Access to the user's account in the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Examples of identifiers in the application:

```
GET /papi/v2/users/1000131/profile HTTP/2
Host: [APPNAME]
Cookie: XSRF-TOKEN=[...]; laravel_session=[...]
X-Xsrf-Token: [...]
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0
Accept: application/json
X-Client_Name-Location: https://[APPNAME]/app/users/1000131
X-Requested-With: XMLHttpRequest
Referer: https://[APPNAME]/app/users/1000131
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

LOCATION

https://[APPNAME]/

RECOMMENDATION

It is recommended to use unpredictable resource identifiers, such as version 4 UUIDs.

More information:

- https://datatracker.ietf.org/doc/html/rfc4122#section-4.4

[INFO] SECURITUM-234069-022: Lack of Rate Limiting

SUMMARY

During the tests, it was found that the API in no way limits the frequency of communication, such as the number of made requests. An attacker exploiting this fact could potentially execute a Denial of Service (DoS) attack, disrupt logic, or cause other security consequences.

More information:

- https://owasp.org/www-project-api-security/

LOCATION

https://[APPNAME]/papi/

RECOMMENDATION

It is recommended to implement the limitation of the frequency of API communication (e.g. HTTP requests) by the client within the specified time frames.