securitum

Security report

SUBJECT

test.bednarek.com.pl web application

DATE

19.04.2023 – 27.04.2023

LOCATION

Poznań (Poland)

AUTHOR

Adam Borczyk

VERSION

1.0

securitum

Executive summary

This document is a summary of work conducted by Securitum. The subject of the test was the web application available at https://test.bednarek.com.pl.

The functionality of making phone calls from the application to a given number was excluded from the scope of the tests.

Tests were conducted using the following roles:

- unauthenticated user,
- user with regular privileges,
- administrator.

The most severe vulnerabilities identified during the assessment were:

- [HIGH] SECURITUM-232444-001: No authentication required,
- [HIGH] SECURITUM-232444-002: Authorization errors lead to privilege escalation.

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

The tests were carried out in the best-effort model. This means placing the main emphasis on critical application functionalities due to the limited duration of security tests.

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, gobuster, Nmap, sqlmap), was used during the assessment.

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.

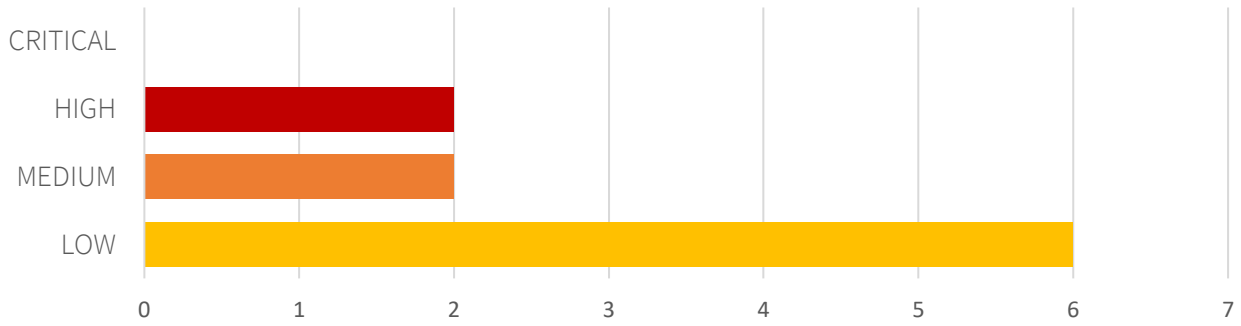
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing a user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser degree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

Security report *1*

Executive summary *2*

Risk classification 3

Statistical overview 3

Change history *6*

Vulnerabilities in the web application *7*

[HIGH] SECURITUM-232444-001: No authentication required 8

[HIGH] SECURITUM-232444-002: Authorization errors lead to privilege escalation 10

Case 1 10

Case 2 11

Case 3 12

[MEDIUM] SECURITUM-232444-003: Lack of protection against Cross-Site Request Forgery (CSRF) attack 14

[MEDIUM] SECURITUM-232444-004: Lack of protection against brute force attack on login form 16

[LOW] SECURITUM-232444-005: Redundant information revealed in detailed error messages .. 18

[LOW] SECURITUM-232444-006: Weak password policy 20

[LOW] SECURITUM-232444-007: Lack of security attributes for sensitive cookie 23

[LOW] SECURITUM-232444-008: Outdated software 25

[LOW] SECURITUM-232444-009: Redundant information disclosure about the application environment in HTTP response headers 26

[LOW] SECURITUM-232444-010: No event logging when performing actions without authentication 27

Informational issues *28*

[INFO] SECURITUM-232444-011: Incorrect parameter values validation 29

[INFO] SECURITUM-232444-012: Lack of Content-Security-Policy header 31

[INFO] SECURITUM-232444-013: Lack of Strict-Transport-Security (HSTS) header 32

[INFO] SECURITUM-232444-014: Lack of Referrer-Policy header 33

[INFO] SECURITUM-232444-015: Lack of protection against Clickjacking attack 34

[INFO] SECURITUM-232444-016: Lack of X-Content-Type-Options header 36

[INFO] SECURITUM-232444-017: Lack of Rate Limiting 37

[INFO] SECURITUM-232444-018: No option to enable 2FA 38

[INFO] SECURITUM-232444-019: Lack of integrity attribute 39

Change history

Document date	Version	Change description
27.04.2023	1.0	Creation of the document.

Vulnerabilities in the web application

[HIGH] SECURITUM-232444-001: No authentication required

SUMMARY

The tested application does not require authentication for access to some functionalities. Thus, any unauthenticated user with network access to the application (i.e. any Internet user) may access data and perform operations that should be reserved only for authenticated users.

By exploiting this vulnerability, it was possible to:

- send text messages (SMS) on behalf of the Client (it was also noticed that the anonymous dispatch event is not logged; details: SECURITUM-232444-010),
- read logs containing recently performed operations in the application,
- read the list of calls made.

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

Knowing the address of a given endpoint and any required parameters.

TECHNICAL DETAILS (PROOF OF CONCEPT)

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

1. Send the following HTTP request to the application, without providing credentials in the `Cookie` header:

```
POST /sms.ashx HTTP/2
Host: test.bednarek.com.pl
Content-Length: 80
Content-Type: application/x-www-form-urlencoded

type=send&telefon=4[...]7&tresc=test-sms-bez-logowania&user_id=308&apis=Serwis
```

2. A message from sender "Serwis" with the content "test-sms-bez-logowania" will be sent to the given phone number.

LOCATION

Global authentication mechanism in the application.

RECOMMENDATION

It is recommended to include in the application the authentication for the above-mentioned functionalities.

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/Authentication_Cheat_Sheet.html

[HIGH] SECURITUM-232444-002: Authorization errors lead to privilege escalation

SUMMARY

The tested application does not implement proper authorization of access to data. This leads to the following problems:

- Case 1: a user can access functionalities reserved for accounts with administrator privileges. By exploiting this vulnerability, it was possible to:
  - display information about other users,
  - create new users and groups,
  - change part of the application configuration,
  - download telephone billings.
- Case 2: a user can perform actions to which he or she has not been granted access through the "uprawnienia" module. Despite setting the selected set of permissions, the application only disables the appropriate fields on the browser side (sets the `disabled` parameter to objects in the DOM tree). On the server side, these permissions are not checked.
- Case 3: a user can perform actions on behalf of another user. To do this, it is enough to know the ID number of the target user. As a result of exploiting the vulnerability, it is possible to send SMS messages on behalf of another user.
  During the audit, there was no knowledge about other functionalities of the application, information from which could be saved in logs. Therefore, it is possible that other places in the application that use the `user_id` parameter are also affected.

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

Case 1 and 2: knowing the address of a given endpoint and any required parameters.

Case 3: an account in the application is sufficient.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Case 1

In order to create new user using an account without privileges to do so, one must:

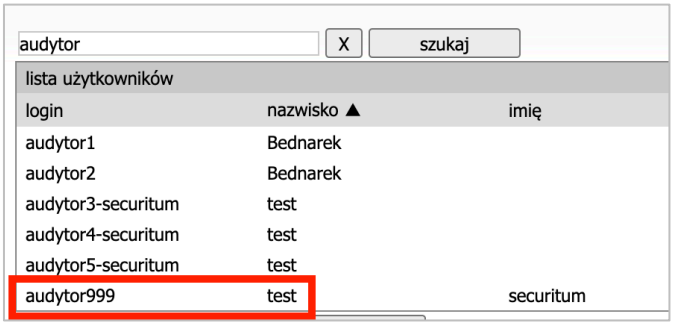
1. Log in to the application using "audytor1" account, which does not have administrative privileges.
2. Send the following HTTP request to the application. The session token which indicates that the "audytor1" account was used, is highlighted in yellow. The newly created account is highlighted in red.

```
POST /users.ashx HTTP/2
Host: test.bednarek.com.pl
Cookie: _callings_przedzi=24.04.2023; ns=audytor1EC[...]F8; 308_callings_przedzi=24.04.2023;
sesja=1; 308_callings_page=1
```

```
Content-Length: 214
Content-Type: application/x-www-form-urlencoded

type=i&nazwa=audytor999&nazwisko=test&imie=securitum&inicjaly=&email=&ip_komputera=&maska=&ip_telefonu=&nr_telefonu=&imei=&autoryzacja_user=1&autoryzacja_ip=0&podsiec=0&edycja_kontrahentow=1&usuwanie_kontrahentow=0
```

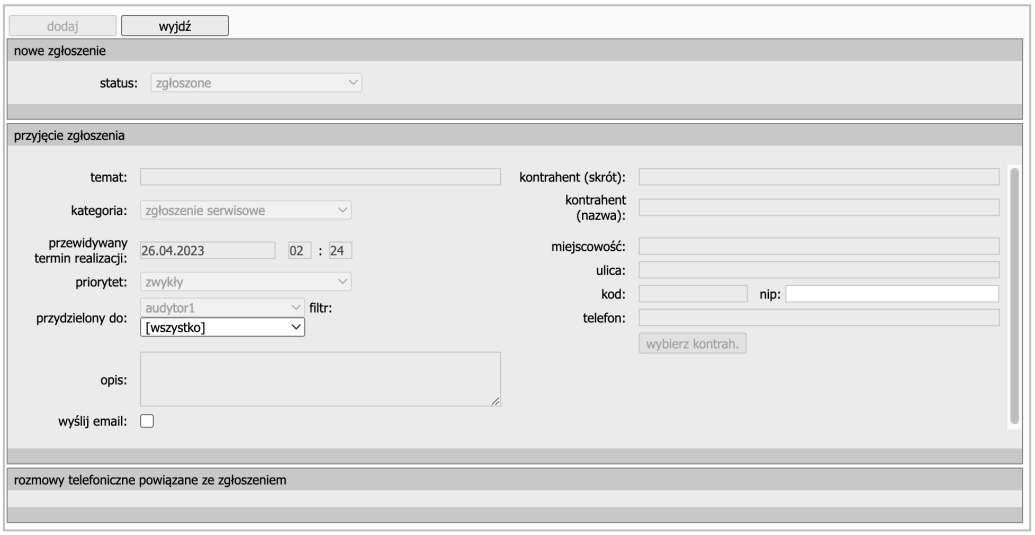
3. In the response body, the server will only return character `1`.
4. Log in using "administrator" account and go to "konfiguracja" tab, and then "użytkownicy".
5. On the users list there is the newly created account with "audytor999" login:



Case 2

In order to gain access to another user's data, the following steps need to be taken:

1. Log in to the application using administrator account.
2. Go to "CRM", and then "uprawnienia" tab.
3. Disable the possibility of creating new orders ("przyj. zlec" group, "C" column) for user "audytor1". Save changes.
4. Log in to the application using "audytor1" account.
5. Go to "zgłoszenia" tab and click "nowe zgłoszenie".
6. Form field with all buttons are visible:



7. Send the following HTTP request to the application. The session token which indicates that the "audytor1" account was used, is highlighted in yellow. The newly created order is highlighted in red.

```
POST /notifications.ashx HTTP/2
Host: test.bednarek.com.pl
```

```
Cookie: _callings_przedzi=25.04.2023; ns=audytor1EC[...]F8; 308_callings_przedzi=25.04.2023;
sesja=1; 308_callings_page=1; 308_notifications_page=1
Content-Length: 560
Content-Type: application/x-www-form-urlencoded

type=i&temat=test-wylaczone-
uprawnienia&firma=&nazwa=&miejscowosc=&kod=&ulica=&telefon=&nip=&powiazanie=0&id_usera=308&id_kat
egorii=3&id_statusu=1&id_priorytetu=3&termin=2023-04-
26%2001%3A49%3A0&id_usera_przydzial=308&id_grupy=308&info=&opis=&id_bilingu=0&email=1
```

8. In response the server will only return value `1`.
9. A new entry with the topic indicated in step 7 will appear on the orders list:

Case 3

In order to send a message on behalf of another user, one must:

1. Log in to the application using “audytor1” account (ID: 308).
2. Send to the application the following HTTP request, which contains “audytor1” user cookie. The ID value of a user account 122, on behalf of which the SMS message will be sent, is highlighted in yellow:

```
POST /sms.ashx 259419 HTTP/2
Host: test.bednarek.com.pl
Cookie: [...];ns=audytor15C[...]12; 308_callings_page=1
Content-Length: 88
Content-Type: application/x-www-form-urlencoded

type=send&telefon=4[...]7&tresc=wyslano%20z%20konta%20audytor1&user_id=122&apis=Serwis
```

3. Log in to the application using “administrator”.
4. Go to “rozmowy”, and then “SMS-y” tab.
5. A message sent by “audytor2” user is seen in the table:

LOCATION

Global verification of user permissions on the server side, regarding all administrator functionalities and permissions granted in the “uprawnienia” module.

RECOMMENDATION

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

In case 3, it is recommended to retrieve information about the user performing the action from his or her session token, instead of from the value of the parameter passed.

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-232444-003: Lack of protection against Cross-Site Request Forgery (CSRF) attack

SUMMARY

The tested application does not implement any protection against Cross-Site Request forgery attack. An attacker may execute any action in the application with another user's privileges, by convincing the user to enter URL, on which malicious HTML/JavaScript code was embedded.

An example of such an action is extending permissions in the application.

The vulnerability is only exploitable when using the Firefox browser. Other leading browsers (Chrome, Edge) have additional protection in the form of automatically assigning the `sameSite=Lax` attribute to session cookies, which means that cookies are not automatically added to HTTP requests initiated from domains other than the application domain.

More details:

- https://sekurak.pl/czym-jest-podatnosc-csrf-cross-site-request-forgery/ (in Polish)
- https://owasp.org/www-community/attacks/csrf
- https://owasp.org/www-project-code-review-guide/reviewing-code-for-csrf-issues
- https://cwe.mitre.org/data/definitions/352.html

PREREQUISITES FOR THE ATTACK

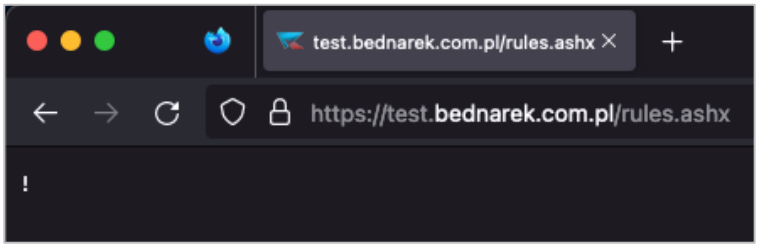
Knowing the structure of the request and persuading selected user to click on the provided link.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Redirecting an administrative user who has an active session in the application and appropriate permissions to the page containing the HTML/JavaScript code given below will result in granting full permissions to create, modify and handle orders. The ID of the user whose privileges will be extended is highlighted in yellow.

```
<html>
  <body>
    <form action="https://test.bednarek.com.pl/rules.ashx" method="POST">
      <input type="hidden" name="308"
value="True&#9;True&#9;True&#9;True&#9;True&#9;True&#9;True&#9;True" />
      <input type="hidden" name="type" value="s" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

As a result of going to a page containing the following code, the server returns a success response (exclamation mark). The screenshot below shows the view in the browser after the execution of the HTML code:



As a result of the request, the “audytor1” user (ID 308) will have a full set of permissions:

LOCATION

All application endpoints that change its state.

RECOMMENDATION

It is recommended for the application to send a random anti-CSRF token in an HTTP response. The token should then be validated on the server side. Requests that do not contain the token or contain it with an incorrect value should be rejected. It is recommended to verify whether the framework used in the application has built-in mechanisms protecting against CSRF.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

[MEDIUM] SECURITUM-232444-004: Lack of protection against brute force attack on login form

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 can perform a brute force attack. 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

None.

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. Enter user login, e.g. "audytor1".
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 an attacker uses an appropriate application (e.g. Burp Suite, Intruder module) or writes a script that will send the request given below.

During the tests, there were 999 failed attempts made to log in to the test account with an incorrect password (highlighted in yellow):

```
POST /login.aspx HTTP/2
Host: test.bednarek.com.pl
Cookie: [...]; ns=
Content-Length: 472
Content-Type: application/x-www-form-urlencoded


__VIEWSTATE[...]&__VIEWSTATEGENERATOR=[...]&__EVENTTARGET=&__EVENTARGUMENT=&__EVENTVALIDATION[...
]&TextBox_name=audytor1&TextBox_pass=test&Button_zaloguj=zaloguj
```

The next request sent with a valid password confirms that the account has not been blocked and presents the ending of enumeration with success:

```
HTTP/2 302 Found
[...]
Set-Cookie: ns=audytor186[...]B8; path=/
Content-Length: 118

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/">here</a>.</h2>
```

```
</body></html>
```

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

LOCATION

Authentication mechanism in the application.

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

[LOW] SECURITUM-232444-005: Redundant information revealed in detailed error messages

SUMMARY

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

More information:

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

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example of a request sent to the application, which refers to a non-working functionality of making phone calls:

```
GET /dial.ashx HTTP/2
Host: test.bednarek.com.pl
```

In response, the application reveals a detailed error message:

```
HTTP/2 500 Internal Server Error
[...]
Content-Length: 4826

<!DOCTYPE html>
[...]
[SqlException (0x80131904): Procedure or function &#39;admin_dial&#39; expects parameter
&#39;@id_uzytkownika&#39;, which was not supplied.]
   DBHelper.Get_sql(String proc, String[] k, String[] v, String cns) in
c:\inetpub\wwwroot\App_Code\DBHelper.cs:97
   dial_Handler.ProcessRequest(HttpContext context) in c:\inetpub\wwwroot\dial.ashx:44
   System.Web.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute() +790
   System.Web.HttpApplication.ExecuteStepImpl(IExecutionStep step) +195
   System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean&amp;
completedSynchronously) +88
</pre></code>

[...]
<!--
Ta strona błędu może zawierać wrażliwe informacje, ponieważ konfiguracja aplikacji ASP.NET dopuszcza
wyświetlanie pełnych komunikatów o błędach za pomocą właściwości &lt;customErrors mode="Off"/&gt;.
Należy rozważyć użycie właściwości &lt;customErrors mode="On"/&gt; lub &lt;customErrors
mode="RemoteOnly"/&gt; w środowiskach produkcyjnych.-->
```

LOCATION

The whole application.

RECOMMENDATION

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

More information:

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

[LOW] SECURITUM-232444-006: Weak password policy

SUMMARY

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

During the audit, it was possible to:

- set a password consisting of only one character,
- set a password consisting of 130 characters, but it was impossible to log back into the account with this password.

In addition, the user is not notified via a separate channel (e.g. e-mail) about the fact that the password was changed.

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 of only one character. The new password is highlighted in yellow in the HTTP request below:

```
POST /pass.aspx HTTP/2
Host: test.bednarek.com.pl
Cookie: [...]
Content-Length: 1829
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4RmYDkLtiMpRJ6Ry
[...]

------WebKitFormBoundary4RmYDkLtiMpRJ6Ry
Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$TextBox_pass"

[...]
------WebKitFormBoundary4RmYDkLtiMpRJ6Ry
Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$TextBox_pass1"

a
------WebKitFormBoundary4RmYDkLtiMpRJ6Ry
Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$TextBox_pass2"

a
------WebKitFormBoundary4RmYDkLtiMpRJ6Ry
Content-Disposition: form-data; name="ctl00$ContentPlaceHolder1$Button_zmien"

zmień hasło
------WebKitFormBoundary4RmYDkLtiMpRJ6Ry—
```

In response, the server accepts the new password:

```
HTTP/2 302 Found
Location: /
Set-Cookie: ns=audytor186[...]B8; path=/
[...]
Content-Length: 118

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/">here</a>.</h2>
</body></html>
```

Then it is possible to correctly log in to this user's account.

LOCATION

User password handling mechanism.

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) 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?");
f) 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;
g) 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);
h) Implementation of an application-based two-factor authentication (2FA), e.g. Google Authenticator (using SMS is absolutely not recommended).

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.

Functionality that will verify the strength of the password should be implemented – 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#implement-proper-password-strength-controls

[LOW] SECURITUM-232444-007: Lack of security attributes for sensitive cookie

SUMMARY

During the audit it was observed that the application did not set security attributes for `ns` session cookie. These attributes are:

- `httpOnly` – informs a browser that the cookie should be available only to the server. Therefore, any attempt to read the cookie from the client side (e.g., by JavaScript) will be blocked. Lack of this attribute could be used by an attacker, e.g., to take over another user's session (when Cross-Site Scripting vulnerability is identified);
- `SameSite` – could prevent browser from sending the cookie between pages with different sites. Implementing this attribute could be helpful, among others, in preventing CSRF attacks. This attack was shown in SECURITUM-232444-003.
- `Secure` – informs a 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.

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
- https://sekurak.pl/flaga-cookies-samesite-jak-dziala-i-przed-czym-zapewnia-ochrone/ (in Polish)

PREREQUISITES FOR THE ATTACK

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

TECHNICAL DETAILS (PROOF OF CONCEPT)

HTTP response to a valid login request, containing the `Set-Cookie` header and cookies without security attributes:

```
HTTP/2 302 Found
Cache-Control: no-store, no-cache, must-revalidate, max-age=0,post-check=0, pre-check=0
Set-Cookie: ns=audytor17E[...]15; path=/
X-Powered-By: ASP.NET
[...]
```

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` – a browser will not send the cookie in cross-site requests,
- `Lax` – a 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.

An example header setting a cookie with security attributes:

```
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-232444-008: Outdated software

SUMMARY

The components of the tested application are:

- jQuery 1.5,
- jQuery UI 1.8.

These are not 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 vulnerabilities, however, the mere fact of using software with publicly known vulnerabilities exhausts the necessity to include such information in the report.

More information:

- https://nvd.nist.gov/vuln/detail/CVE-2011-4969
- https://nvd.nist.gov/vuln/detail/CVE-2022-31160
- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

PREREQUISITES FOR THE ATTACK

Dependent on vulnerability.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example request:

```
GET /planning_data.html HTTP/2
Host: test.bednarek.com.pl
```

The HTTP response shows an outdated version of the software being used:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>
```

LOCATION

https://test.bednarek.com.pl/planning_data.html

RECOMMENDATION

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

[LOW] SECURITUM-232444-009: Redundant information disclosure about the application environment in HTTP response headers

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

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example response with redundant headers:

```
HTTP/2 200 OK
Cache-Control: private
Content-Type: text/plain; charset=utf-8
Server: Microsoft-IIS/10.0
X-Aspnet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 25 Apr 2023 13:58:07 GMT
Content-Length: 1

1
```

LOCATION

The whole application.

RECOMMENDATION

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

[LOW] SECURITUM-232444-010: No event logging when performing actions without authentication

SUMMARY

It has been observed that the application has the functionality of logging SMS messages sent from the application, which is available to users with administrative privileges. The message sending event is saved in the operation list with date, content and author. When using the SECURITUM-232444-001 vulnerability that allows sending without the need for authentication, it was noticed that such an event is not logged.

PREREQUISITES FOR THE ATTACK

None.

TECHNICAL DETAILS (PROOF OF CONCEPT)

In order to observe the lack of event logging in the absence of authentication, it is necessary to:

1. Make the following HTTP request (without the Cookie header) using the SECURITUM-232444-001 vulnerability:

```
POST /sms.ashx HTTP/2
Host: test.bednarek.com.pl
Content-Length: 74
Content-Type: application/x-www-form-urlencoded

type=send&telefon=4[...]7&tresc=sms%20bez%20cookie&user_id=308&apis=Test
```

2. Log in to the application using “administrator” account.
3. Go to “rozmowy”, and then “SMS-y” tab.
4. The message sent in point 1 is not on the list.

LOCATION

Authentication mechanism and event log handling.

RECOMMENDATION

It is recommended to analyse whether the functionality of saving events performed by users works correctly, i.e. whether it is triggered at the right time in relation to the application logic or whether it has no errors that violate the transactionality of the action-logging pair.

Informational issues

[INFO] SECURITUM-232444-011: Incorrect parameter values validation

SUMMARY

It was observed that the application incorrectly parses data entered. By entering data with a newline, it is possible to inject additional parameters into future HTTP requests of other users.

This is due to the data structure used, which has an exemplary form:

```
!
3       zgłoszenie serwisowe  0
25      test-scrt 1
26      test2 3
```

It is interpreted as plain text. As a consequence, the newline character will disturb this structure by adding a new line with incomplete parameter contents. A browser, constructing data for subsequent HTTP requests, may misinterpret them, creating additional and/or removing other existing parameters.

During the audit it was not possible to exploit this behaviour in the context of an application security breach, however, it is recommended to use standardised data structures, such as JSON or XML, to prevent possible injections.

TECHNICAL DETAILS (PROOF OF CONCEPT)

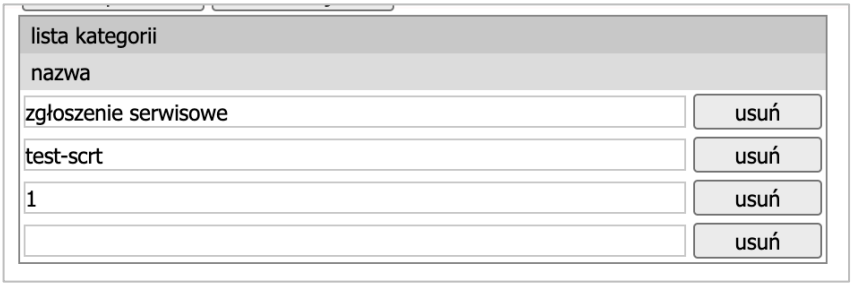
In order to confirm the above-mentioned behaviour, one must:

1. Configure a browser to route traffic through a proxy tool, e.g. Burp Suite.
2. Log in to the application using “administrator” account.
3. Go to “CRM, and then “słowniki” and “kategorie” tab.
4. Add a new category called `test-scrt` and click “zapisz”, and intercept the request, in order to modify it in transit.
5. Modify `test-scrt` value as shown below:

```
POST /categories.ashx HTTP/2
Host: test.bednarek.com.pl
Cookie: ns=audytor186[...]B8
Content-Length: 38
Content-Type: application/x-www-form-urlencoded

i=test-scrt%0anowyparametr=1234&type=s
```

6. Forward the request.
7. Refresh categories list page. The table will look as follows (note the additional entry called “1”):



The server response will show the following data with a disturbed structure:

```
!
3       zgłoszenie serwisowe  0
27      test-scrt
nowyparametr=1234     1
```

8. Without making any changes, click the “zapisz” button.
9. The browser will send the request below. An additional `nowyparametr=1234=1%092` parameter is present in the request:

```
POST /categories.ashx HTTP/2
Host: test.bednarek.com.pl
Cookie: ns=administrator9E[...]08
Content-Length: 351
Content-Type: application/x-www-form-urlencoded
[...]

27=test-
scrt%091&nowyparametr=1234=1%092&i=%093&type=s&toJSONString=function%20(filter)%20%7B%0A%20%20%
20%20%20%20%20%20%20%20%20%20return%20JSON.stringify(this%2C%20filter)%3B%0A%20%20%20%20%20%20%2
0%7D&parseJSON=function%20(filter)%20%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20return%20JSON.pars
e(this%2C%20filter)%3B%0A%20%20%20%20%20%20%20%20%7D
```

LOCATION

All user input, returned in plain text.

RECOMMENDATION

It is recommended to limit the possibility of entering special characters that do not match the expected data type. In addition, it is recommended to encode special characters when returning them from the server to the client.

[INFO] SECURITUM-232444-012: 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 a 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://sekurak.pl/wszystko-o-csp-2-0-content-security-policy-jako-uniwersalny-straznik-bezpieczenstwa-aplikacji-webowej/ (in Polish)
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

LOCATION

The whole application.

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-232444-013: 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
- https://sekurak.pl/hsts-czyli-http-strict-transport-security/ (in Polish)

LOCATION

The whole application.

RECOMMENDATION

The server 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-232444-014: 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

The whole application.

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-232444-015: Lack of protection against Clickjacking attack

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 a victim of the attack to perform an unauthorized operation.

More details:

- https://owasp.org/www-community/attacks/Clickjacking
- https://owasp.org/www-community/attacks/Cross_Frame_Scripting
- https://sekurak.pl/nietypowe-metody-wykorzystywane-w-atakach-phishingowych/ (in Polish)

PREREQUISITES FOR THE ATTACK

The victim performs some action on a webpage prepared by the attacker.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Sample request sent to the application confirming the existence of the issue:

```
GET /login.aspx HTTP/2
Host: test.bednarek.com.pl
```

In response, the application reveals the missing “X-Frame-Options” header:

```
HTTP/2 200 OK
Cache-Control: no-store, no-cache, must-revalidate, max-age=0,post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html; charset=utf-8
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Server: Microsoft-IIS/10.0
P3p: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
X-Ua-Compatible: IE=7
X-Aspnet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 26 Apr 2023 13:14:01 GMT
Content-Length: 3277

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
[...]
```

LOCATION

The whole application.

RECOMMENDATION

It is recommended that the application sets the "Content-Security-Policy" header for each page, and in it the `frame-ancestors` directive by selecting one of the following options:

a) Complete blocking of the page in a frame:

```
Content-Security-Policy: frame-ancestors 'none'
```

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

```
Content-Security-Policy: frame-ancestors 'self'
```

It is also possible to use the "X-Frame-Options" header, however, this form of protection against placing the application in an iframe is considered obsolete and should only be used to provide support for old versions of browsers. The "X-Frame-Options" header should have of the following options:

a) Complete blocking of the page in a frame:

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

b) Possibility to place a page in a 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/Content-Security-Policy/frame-ancestors
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

[INFO] SECURITUM-232444-016: 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 response received, 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.

Implementing this header is considered a general good hardening practice for web applications.

More information:

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

LOCATION

The whole application.

RECOMMENDATION

The following header should be added in all server responses:

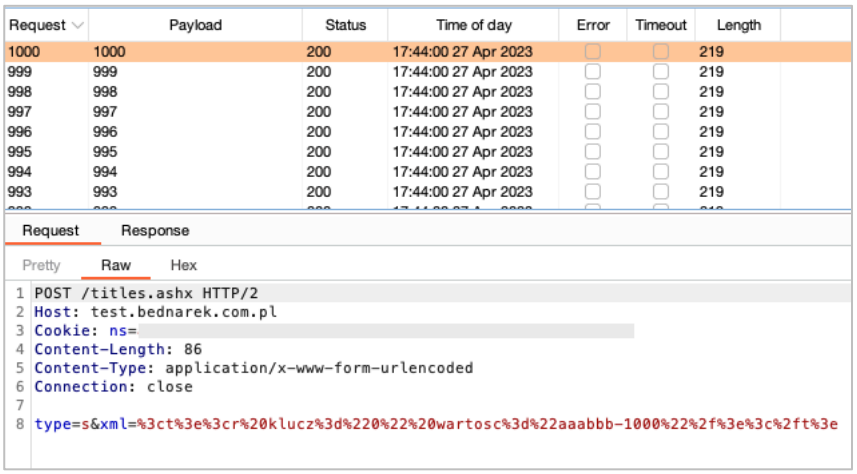
```
X-Content-Type-Options: nosniff
```

[INFO] SECURITUM-232444-017: 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 requests made. An attacker exploiting this fact could potentially execute a Denial of Service (DoS) attack, disrupt logic, or cause other security consequences.

During the audit, 1,000 requests to the "/titles.aspx" endpoint were made within 6 seconds, which required the server to parse data in XML format and save them in the database. All were successful, and the server was not found to delay client requests in any way:



Request	Payload	Status	Time of day	Error	Timeout	Length
1000	1000	200	17:44:00 27 Apr 2023			219
999	999	200	17:44:00 27 Apr 2023			219
998	998	200	17:44:00 27 Apr 2023			219
997	997	200	17:44:00 27 Apr 2023			219
996	996	200	17:44:00 27 Apr 2023			219
995	995	200	17:44:00 27 Apr 2023			219
994	994	200	17:44:00 27 Apr 2023			219
993	993	200	17:44:00 27 Apr 2023			219

Request Response

Pretty Raw Hex

```
1 POST /titles.ashx HTTP/2
2 Host: test.bednarek.com.pl
3 Cookie: ns=
4 Content-Length: 86
5 Content-Type: application/x-www-form-urlencoded
6 Connection: close
7
8 type=s&xml=%3ct%3e%3cr%20klucz%3d%220%22%20wartosc%3d%22aaabbb-1000%22%2f%3e%3c%2ft%3e
```

More information:

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

LOCATION

The whole application.

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.

More information:

- https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa4-lack-of-resources-and-rate-limiting.md

[INFO] SECURITUM-232444-018: No option to enable 2FA

SUMMARY

During the tests, the possibility of securing the account with the use of Two-Factor-Authentication (2FA) was not found. This is a mechanism that adds an additional layer of security to the account – during the login process, after entering the correct login details, a user is asked to provide a Time-Based One-Time Password (TOTP) in the form of several digits (or characters) generated by, e.g. mobile application. Accounts configured with 2FA offer a greater level of security against dictionary/brute-force attacks. In addition, if an attacker obtains the login details of the victim's account, e.g. by leaking a password in another, independent system, he or she will not be able to log in to the user's account without the 2FA code.

RECOMMENDATION

It is recommended to implement 2FA as an optional feature that can be enabled at user's request.

[INFO] SECURITUM-232444-019: Lack of integrity attribute

SUMMARY

The application loads and executes external scripts of the third parties.

However, it is not verified that the requested file has the correct checksum. This means that an attacker can swap the content of scripts on a third-party server, which will later launch malicious scripts in the application.

Adding the `integrity` attribute in the `<script>` elements allows to enable an additional mechanism to protect against the above scenario: before the script is executed, a browser will check if its checksum is as it should be. In case the checksums do not match, the script will not be executed.

More information:

- https://www.w3.org/TR/SRI/

PREREQUISITES FOR THE ATTACK

Swapping the content of a script on a third-party server.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Below, there is an example of loading an external script without checking the checksum, in response to a request to `/planning_data.html` path:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>
```

LOCATION

https://test.bednarek.com.pl/planning_data.html

RECOMMENDATION

It is recommended to set the `integrity` HTML attribute when referring to external resources.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html#subresource-integrity