Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · [redacted]

Pentest-Report nthLink VPN Apps 06.-07.2019

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. A. Aranguren (ext.)

Index

Introduction

"The Internet has permanently altered how we communicate and live. Advancements in technology have given us immediate access to information with a few clicks and taps. However, an alarming number of authoritarian governments have imposed tighter control over the free flow of information with invasive technologies.

nthLink is a powerful anti-censorship mobile application capable of circumventing Internet censorship and self-recovering from blocking events. Most importantly, it incorporates strong encryption to protect the information flow between the consumer and the source."

From https://www.nthlink.com/

This report documents the findings of a large-scale security assessment of the nthLink VPN applications. Carried out by Cure53 in June 2019, the project entailed both a penetration test and a source code audit. It specifically targeted two mobile applications that are supposed to provide users with secure VPN services by relying on Outline VPN. As a result of the investigation, seven security-relevant flaws were spotted on the scope.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · [redacted]

To give some context, Cure53 executed this assessment on request from the U.S. Agency for Global Media and the examined apps were built by Advanced Circuiting Inc. (ACI), which also managed the technical parts of the audit after initial preparations. Corresponding to the project's objectives, Cure53 comprised a team of two testers and allocated a time budget of six days to this project. As for the specific timeline, the work was initiated in early June 2019 and finalized - with this report - in early July of the same year.

For this assessment, Cure53 could take advantage of the so-called white-box methodology. This approach means that the testing team was granted access to all relevant sources as well as test-builds and debug versions of the examined apps. Over the course of the engagement, Cure53 communicated with the ACI managing team via email and all these exchanges were prompt and productive. The ACI team was very helpful and supported Cure53 in accomplishing the level of coverage that was expected by the involved parties.

It should be clarified that the goals of the audit concerned reaching the best possible coverage and performing deep-dive-style investigations into security and privacy related matters relevant for the nthLink VPN apps. Under this premise, Cure53 identified three vulnerabilities and four general weaknesses (with lower exploitability potential) on the scope. One item was ascribed with a "*High*"-severity marker.

Elaborating on the above, it needs to be noted that a rather strong attacker model was assumed for this project. This was due to the high probability that the app would be employed by users who might not be able to trust the network they rely on for Internet access. Further, an attacker who seeks to extract local data from the phones on which the apps are installed was also considered. Similarly, Cure53 also investigated what capabilities would be available to an attacker who is simply interested in having the app stop delivering the services it is expected to furnish. The above clarifications are needed because they formed a basis for how the severity of the tracked issues was calculated. To reiterate, one of the seven problems carried *"High"* risk and two other vulnerabilities were deemed to pose *"Medium"*-level threats. The remaining findings are rather trivial and could only be applied with "*Low*" and *"Informational"* scores.

In the following sections, this report will first shed light on the scope and then furnishes case-by-case descriptions of the findings, featuring both technical details and possible mitigation for going forward. Based on the results of this summer 2019 assessment, Cure53 issues a broader verdict about the privacy and security posture of the tested items. Conclusions pertinent to the nthLink VPN applications being ready for production usage by the targeted audiences are supplied in the final section of this document.

Scope

- **nthLink VPN apps using Outline**
    - https://nthlink.com/android
    - https://nthlink.com/ios
- Debug-ready app-builds were made available to Cure53.
- Sources were provided to Cure53

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *NTH-01-001*) for the purpose of facilitating any future follow-up correspondence.

NTH-01-001 Android/iOS: No Pinning on the server provisioning traffic *(Medium)*

The implementations of the tested VPN solution on Android and iOS were found to provision the VPN server configuration over TLS, yet without the accurate Pinning protection in place. This may be an issue for users who presume that the VPN will protect them against high-profile attackers able to forge a certificate that is trusted by the Android/iOS certificate store. A high-profile attacker, such as a governmental agency, could leverage this weakness to modify the HTTP response so that the users would be connecting to an attacker-controlled VPN server instead of the intended one.

This issue was confirmed by monitoring HTTP traffic at runtime during testing. The observed HTTPS requests are provided next.

Android:

URL:
https://www.dfc4819385ef.info/getserver/iJkeoPdug7dhaCde

Response:
```
HTTP/1.1 200 OK
[...]

{
  "host": "3.91.12.18",
  "method": "chacha20-ietf-poly1305",
  "name": "nthLink Server",
  "password": "CA3Rksjh9NCY",
  "port": 3598
}
```

iOS:

URL:
https://www.03fc33fc0851.info/getserver/iJkeoPdug7dhaCde

Response:
```
HTTP/1.1 200 OK
[...]

{
  "host": "3.91.12.18",
  "method": "chacha20-ietf-poly1305",
  "name": "nthLink Server",
  "password": "CA3Rksjh9NCY",
  "port": 3598
}
```

In order to eliminate this issue, it is recommended to implement Pinning on Cordova Android and iOS apps. This can be achieved by using the *Cordova Advanced HTTP* plugin[1]. Please note that although the number of domains is high, the certificates could be self-signed and, additionally, all of them might be signed with a *root* CA. The clients could then implement Pinning and verify the signature of the *root* CA instead of checking the certificates linked to individual domains.

NTH-01-005 Android: Possible VPN takeover via domain registration *(High)*

It was found that the VPN server provisioning mechanism on Android is based on a Domain Generation Algorithm (DGA) which, in turn, relies on the date of the given day. This mechanism is easily reversible in the decompiled APK. Thus, a malicious attacker could register some of the future domains before nthLink does, eventually making nthLink users connect to attacker-controlled VPN servers.

This issue can be observed in the server configuration retrieval implementation, as found on the decompiled Android APK provided for testing. It is also possible to confirm this problem at runtime by deleting the *org.outline.vpn.VpnConnectionStore.xml* file and seeing how the app will query the same server for a given day, but looks up a different one the next day.

[1] https://github.com/silkimen/cordova-plugin-advanced-http

Files:
/assets/www/cordova_main.js
/assets/www/app/app - 副本.js
/assets/www/app/app.js

Affected Code:
```
App.prototype.getRequestUrl = function () {
   var seed = "A north sketch ducks below the acid.";
   var md5 = crypto.createHash('md5');
   var date = new Date().toISOString().split('T')[0];
   var domain = md5.update("" + seed + date).digest('hex');
   return "https://www." + domain.substring(0, 12) +
".info/getserver/iJkeoPdug7dhaCde";
    };
    App.prototype.getServerConfig = function () {
[...]
     xhr.open('GET', _this.getRequestUrl(), true);
```

It is recommended to implement a signature that clients can validate using an nthLink public key. This will make it possible for clients to ensure that the VPN connection details can be trusted. The signature could be implemented as an extra field in the returned VPN server configuration. An alternative approach would be to protect TLS communications to these provisioning servers with Pinning, as explained in NTH-01-001.

NTH-01-006 iOS: VPN DoS via single point of failure *(Medium)*

The provided source code and the iOS app use a single domain for the VPN server provisioning. This makes it trivial for a censor to block the IP addresses of the relevant domain's name, hence preventing users from relying on the VPN service. This issue can be confirmed by looking at either the provided source code or, more specifically, by studying the source code of the iOS application on any iOS device.

Files:
/private/var/containers/Bundle/Application/…./Outline.app/www/cordova_main.js [iOS]
/private/var/containers/Bundle/Application/…./Outline.app/www/app/app.js [iOS]
src/nthlink-master/src/www/app/app.ts [source code provided for this audit]

Affected Code:
```
App.prototype.getServerConfig = function () {
       return new Promise(function (resolve, reject) {
              var xhr = new XMLHttpRequest();
              xhr.onload = function () {
                     try {
                            var json = JSON.parse(xhr.responseText);
                            if (!json['method'] || !json['password'] ||
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · [redacted]

```
                            !json['host'] || !json['port']
                            || !json['name']) {
                            return reject(
                                  new Error('Incorrect server config'));
                     resolve(json);
              } catch (err) {
                     reject(err);

       };
       xhr.open('GET',
              "https://www.03fc33fc0851.info/getserver/iJkeoPdug7dhaCde",
       true);
       xhr.setRequestHeader("Content-Type", "application/json");
       xhr.setRequestHeader("Accept", "application/json");
       xhr.send();
  });
};
```

It is recommended to implement a more sophisticated mechanism for the VPN provisioning. An interesting approach in this area would be an implementation of *Psiphon*[2], wherein clients discover VPN servers slowly. As such, blocking is rendered substantially more difficult for censors. Please note that a *Psiphon Cordova Plugin*[3] could be used directly or consulted as a reference on how to improve the VPN server provisioning implementation.

[2] https://psiphon.ca/en/faq.html
[3] https://www.npmjs.com/package/psiphon-cordova-plugin

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

NTH-01-002 Android: VPN credentials leaking via Android backups *(Low)*

It was found that the *backup* element is not specified on the *Android Manifest*. Therefore, on the Android app, it defaults to *true*. In rare scenarios where the user has USB debugging enabled, this could result in illegitimate access to the VPN credentials, as these are stored in plain-text on the app's shared preferences.

The issue can be confirmed by observing the following file, which is created when the user connects to the VPN for the first time.

File:
org.outline.vpn.VpnConnectionStore.xml

Contents (decoded):
```
connection,{"id":"5a3f5380-ee89-4b01-bd9a-1f703fe82f32","config":
{"host":"3.91.12.18","method":"chacha20-ietf-poly1305","name":"nthLink
Server","password":"CA3Rksjh9NCY","port":3598}}
```

It is recommended to explicitly disable backups in the *Android Manifest* to eliminate this attack vector. In addition, the VPN passwords could be stored in a more secure way with the use of the Android KeyStore. In Cordova apps, this can be accomplished via the *SecureStorage plugin*[4]. Please note that following this advice will also leverage the iOS keychain and results in general storing of secrets on that platform occurring in a safer way.

NTH-01-003 iOS: Data leakage due to missing *Data Protection* entitlement *(Info)*

It was found that the iOS app fails to leverage the native iOS *Data Protection* for stronger encryption of data files. This means all files remain unprotected and readable while the phone is locked. A malicious attacker could leverage this weakness to retrieve the VPN connection details from the application files while the phone is locked, as the decryption key for these files remains in memory.

This issue was confirmed by running the *tar* command while the phone was locked. This was done from the *application data* directory and - as the output illustrates - 62

[4] https://www.npmjs.com/package/cordova-plugin-secure-storage

application files (which means nearly all files) are unprotected at present. This strongly suggests that the *Data Protection* entitlement has not been enabled. This appears to neither be the case at the app level, nor at the individual level of the files.

Command:
```
tar cvfz files_locked.tar.gz * > output.txt
```

Output:
```
tar: Library/Caches/Snapshots/com.nthlink.ios.client/2E5176A0-8B22-4E57-ABE9-
6ACDDE4E4407@2x.ktx: Cannot open: Operation not permitted
tar: Library/Caches/Snapshots/com.nthlink.ios.client/downscaled/B8F2F003-FE78-
4862-9AFA-C2FBF0156BF5@2x.ktx: Cannot open: Operation not permitted
tar: Exiting with failure status due to previous errors
```

Command:
```
wc -l output.txt
```

Output:
```
62 output.txt
```

From here, it was found that the VPN credentials can be read in clear-text from the *NSURLCache.*

File:
Library/Caches/com.nthlink.ios.client/nsurlcache/Cache.db

Table:
cfurl_cache_receiver_data

Contents (column: *receiver_data*):
```
{
"host": "3.91.12.18",
"method": "chacha20-ietf-poly1305",
"name": "nthLink Server",
"password": "CA3Rksjh9NCY",
"port": 3598
}
```

It is recommended to turn the *Data Protection Capability* on at the application-level. This can be trivially accomplished from the *"Capabilities"* tab in *XCode* and will automatically protect all files in the application. Alternatively, the file encryption process could be implemented on a file-by-file basis as well[5]. The easiest way to accomplish this would be

[5] https://developer.apple.com/documentation/uikit/protecting_the_user_s_privac...r_app_s_files

to open the Cordova for iOS project in XCode[6] and enable "*Data Protection"* on the *"Capabilities"* tab.

NTH-01-004 Android/iOS: Cordova configuration allows unrestricted access *(Info)*

It was found that the Android and iOS apps, as well as the source code, deploy a Cordova configuration file that allows links to open arbitrary URLs. Further, these permit the Cordova WebView to make requests without restrictions. While the application simply establishes a VPN connection at the moment, this might become an issue in future releases, for example if an XSS vulnerability is introduced.

This issue can be confirmed by looking at the *config.xml* file on the source code of the Android or iOS apps.

File:
config.xml

Affected Code:
```
<content src="cordova_index.html" />
<access origin="*" />
<allow-intent href="http://*/*" />
<allow-intent href="https://*/*" />
```

It is recommended to implement a whitelist of the allowed origins and URLs wherever possible. This will significantly reduce the potential impact of XSS vulnerabilities on the application in the future. Detailed information on achieving this, including CSP guidance, can be found on the documentation of the *Cordova Whitelist Plugin*[7].

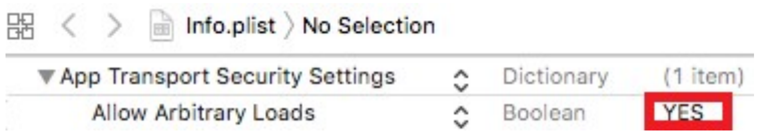
NTH-01-007 iOS: Weakened ATS configuration allows clear-text HTTP traffic *(Info)*

It was found that the iOS app weakens the default iOS configuration by turning off *App Transport Security (*ATS). This allows the app to perform insecure clear-text HTTP requests, which are otherwise not generally allowed on iOS. Please note that this weakness is not mitigated by the Cordova configuration of the app, since it explicitly permits clear-text HTTP connections as well (see NTH-01-004).

This issue can be confirmed by observing the "*Allow Arbitrary Loads*" value of the *Info.plist* file after installing the iOS app on a Jailbroken device.

[6] https://cordova.apache.org/docs/en/latest/guide/platforms/ios/
[7] https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · [redacted]



```
▼ App Transport Security Settings    Dictionary   (1 item)
    Allow Arbitrary Loads            Boolean      YES
```

Fig.: ATS Configuration weakening on Info.plist

According to the official iOS documentation, making the ATS configuration weaker requires supplying a justification during the App Store review. In particular, the documentation[8] states:

*"A Boolean value indicating whether App Transport **Security restrictions are disabled for all network connections**.*
[...]
*Set this key's value to **YES to disable App Transport Security (ATS) restrictions for all domains** not specified in the NSExceptionDomains dictionary. [...]*

Important

***You must supply a justification during App Store review if you set the key's value to YES**, as described in Provide Justification for Exceptions. Use this key with caution because it significantly reduces the security of your app. In most cases, it's better to upgrade your servers to meet the requirements imposed by ATS, or at least to use a narrower exception."*

It is recommended to stick to the iOS defaults and not weaken the ATS configuration without reasons. Furthermore, the Cordova configuration should also disallow clear-text HTTP requests, as explained in NTH-01-004.

[8] https://developer.apple.com/documentation/bundleresources/inform...curity/nsallowsarbitraryloads

Conclusions

The results of this June 2019 Cure53 assessment of the two nthLink VPN mobile applications, which were placed in scope by the U.S. Agency for Global Media and the ACI team, testify that the examined items are quite sound from a security perspective. At the same time, there is some room for improvement that two members of the Cure53 team completing this project could identify during this engagement. Specifically, despite the applications being simple and not offering substantial attack surface in the first place, the auditors managed to spot seven security-relevant flaws.

In particular, after spending six days on testing the apps, Cure53 needs to emphasize that "dead" code is a weak point of the developed products. For example, the applications no longer support *invites* in theory, yet it was found that the invite code remains reachable - either via intents on Android or with a custom URL handler on iOS. Similarly, the flaw also affects the clipboard. Even though this suggests suboptimal approaches in place, Cure53 did not spot any vulnerability in this realm on this occasion.

Moving on two the array of seven actual discoveries stemming from this test, it should be underscored that all three vulnerabilities are connected to the VPN server's faulty provisioning implementation. In other words, the problems negatively impact a mechanism that provides VPN server details to the mobile apps prior to connecting to the VPN via *Shadowsocks*. The highest severity finding - noted as "*High*"-ranking risk, showcases the issue in the Domain Generation Algorithm (DGA). This dictates which server the app will connect to each day and illustrates retrieval of the configuration pertinent to the VPN server used for connections. This algorithm was found to be date-based and easily reversible from the Android APK. Malicious attackers could run the same algorithm locally to predict future domains and register them earlier than the nthLink complex could do so. This way, taking over VPN users by connecting them to attacker-controlled servers could be accomplished.

On a positive note, no VPN leaks could be identified during this project and, more broadly, the apps simply constitute a UI on top of a mature, more than stable and widely used SOCKS/VPN solution known as *Shadowsocks*. Leveraging mature solutions and libraries instead of rolling a custom solution from scratch usually tends to be an excellent choice. In this case, the lack of findings in this report in regard to the VPN leaks confirms this in full.

All in all, the apps should be seen as relatively solid, especially as they offer little opportunity for a compromise thanks to the well-chosen and correctly deployed libraries and frameworks. The nthLink VPN scope examined during this June 2019 project exposes a small attack surface and is marked by a reduced potential for security mistakes happening in the applications' implementation during the development process. Once all issues listed in this Cure53 report have either been addressed properly or are flagged as acceptable risks (severity permitting), the tested applications should be considered production-ready.

Cure53 would like to thank [redacted] [redacted] for their excellent project coordination, support and assistance, both before and during this assignment.

Addendum A

nthLink Security Remediation Plan

The nthLink Security Remediation Plan addresses the issues in Cure53's security audit.

nthLink Security Remediation Plan

For 2019 Cure53 Security Audit

NTH-01-005 Android: Possible VPN take over via Domain Registration (High)

Status: Fixed

Action/Plan

nthLink uses a time-based domain generation algorithm to determine the directory server's domain name at any given time. To prevent attackers from taking over domain registration for future domain names, nthLink utilizes the following preventative methods.

- nthLink's directory servers sign VPN configuration information returned to the client apps. Both the iOS and Android apps now check server signature before accepting new VPN configurations.
- In production mode, nthLink will pre-register domain names several months ahead.
- nthLink will change the Domain Generation algorithm's seed in each major release.
- nthLink uses different seed for iOS and Android.
- In future implementations, we will maintain a seed pool and each user will receive a subset of the seeds. Each user's seed will be updated dynamically without the need to download/install a new app.

NTH-01-001 Android/iOS: Lack of Pinning on Server Provisioning Traffic (Medium)

Status: Fixed

Action/Plan

nthLink's directory servers now sign VPN configuration information returned to the client apps. Both the iOS and Android apps now check server signature before accepting new VPN configurations.

Following is an excerpt of the codes added for checking server signature in *app.ts*.

```
private readonly publicKey: string = "-----BEGIN PUBLIC KEY-----
\nMIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA9/W/2SBYDG9rlQ3XJt39\np2ebGsZo81o1Oq6cPwP0BH
IvfjeWf3l0fQaNS1zAgTenyBWNxV4Sk526mGFnnpeP\n2Fjx6YMsIdULSFoz63is1Inii82DGLE5CWvzM1RvZkV8rQ
5UcWRPh3je2g6Vzyd0\nAKA0xxTqvQQbnsK1sEK9biMI2242yvzUEOI36M9dVr5WOzZurIC+RgE4OjAsfGNc\n5rNu
2ILO+T0Zq5YOiOaqh1CmvlVwlazvjUcdsEPitsMi01w4DLdAi8qJFO1dNNaE\njDFMVXT5Sxk/lmpoeRzG+aYBnd3L
lIDlaaSG1ja0gxf8GHoqckLAiiV8OyDJA5Jn\nySGh0rjkuUkncmhAyrK6bEFnQYhaqxXEEUTikKhYFi0A/17JOkRX
yOW/uNhS3lQo\nZ42GkYlAaKSqFR4TA6nNmpup3eTyGpUKwjZqy37PT8SKytD9I1yM3No5KvtSV/lh\n05yf0+JJZL
0a4ChDLWa0OEuuaY/ocKO4VuVB+3KpbgfF8uAOvGBMk60QUGoG6vDK\njm2TIzxYCWojihmThx319mFytovJd/JP/c
8vXVvDO4fJOYMbPhjYMju8/HmH2atE\nW1dgnzDHpO9ngALzJ8XM94V0DGPvqqKg/UqOCCYZy9Zc4YofE34/7tIicI
/ho4Kw\nzMZ1ek4b30+kpMJ/b0xQ0UkCAwEAAQ==\n-----END PUBLIC KEY-----";

//...

private decrypt(data: string): string {
    try {
        const arr = data.split('*-*');
        if (arr.length < 2) {
            console.warn('Invalid data');
            return data;
        }
        const signature = arr[0];
        const cipher = arr[1];
        const verifier = crypto.createVerify('RSA-SHA256');
        verifier.update(cipher);
        const ver = verifier.verify(this.publicKey, signature, 'base64');
        if (!ver) {
            console.warn('Invalid signature');
            return data;
        }
        else {
            const ivHex = cipher.substr(0, 32);
            const decryptIv = Buffer.from(ivHex, "hex");
            const data = cipher.substr(32);
            const decipher = crypto.createDecipheriv("aes-256-ctr",
                this.jsonSeed, decryptIv);
            const decrypted = decipher.update(data, "hex", "utf8") +
                decipher.final('utf8');
            return decrypted;
        }
    } catch (exception) {
        console.warn('cannot decrypt the data.');
        return data;
    }
}
```

NTH-01-006 iOS: VPN DoS via Single Point of Failure (Medium)

Status: Investigating (will implement in future releases)

Action/Plan

Per Cure53’s recommendation, nthLink will implement a more sophisticated VPN provisioning algorithm in future releases, which the apps will discover both Domain Generation seeds and

VPN configurations gradually. In addition, each nthLink user will receive only a subset of the Domain Generation seeds and VPN configurations depending on multiple factors including location, IP address, and platform, etc.

NTH-01-002 Android: VPN Credential Leakage via Android Backups (Low)

Status: Planned (will fix in the next release)

Action/Plan

Per Cure53's recommendation, nthLink will explicitly disable backups on the Android Manifest to eliminate this attack vector in the next release. In addition, nthLink will store VPN configurations using the Android KeyStore through Cordova's SecureStorage plugin.

NTH-01-003 iOS: Data Leakage via Lack of Data Protection Entitlement (Info)

Status: Planned (will fix in the next release)

Action/Plan

Per Cure53's recommendation, nthLink will turn on the Data Protection Capability at the app level in the next release.

NTH-01-004 Android/iOS: Cordova Configuration allows Unrestricted Access (Info)

Status: Planned (will fix in the next release)

Action/Plan

Per Cure53's recommendation, nthLink will implement a white-list of allowed origins and URLs to reduce the potential impact of XSS vulnerabilities against the application in the next release.

NTH-01-007 iOS: Weakened ATS Config allows clear-text HTTP Traffic (Info)

Status: Planned (will fix in the next release)

Action/Plan

Per Cure53’s recommendation, nthLink will use iOS defaults and not weaken the ATS configuration. In addition to this, the Cordova configuration will also disallow clear-text HTTP requests per Cure53’s notes in NTH-01-004.

Addendum B

Remediation Plan Verification

Cure53 reviewed the nthLink Security Remediation Plan and concluded that Advanced Circuiting Inc has addressed all the issues accordingly.