Threat Modeling for Nahoft

Title: Threat Modeling for Nahoft
Authors: U4I
Status: Draft
Last update: Oct 2021 (v 0.9)
==========

Nahoft and its Motivation

Nahoft is an asymmetric encryption tool that allows users to encrypt any text message under 1,000 characters on Android phones (Android 8.0 and above) before sharing, to add an extra layer of security against Man-in-the-middle (MITM) attacks.

Nahoft can encrypt your text message as a photo (known as steganography) or as a "string of Persian words" chosen from a list of 400 common and innocuous Persian words.

Nahoft was created with Iran and its unique political and socio-economic situation in mind. To communicate during an Internet shutdown, users would enter messages into Nahoft and the app would give you an encrypted message that then can be communicated through less secure means such as Iran's National Information Network (NIN), making it very difficult for the adversaries to compromise the confidentiality of the messages.

*This document outlines the threat models of users/situations who would benefit from using Nahoft, and threat models that Nahoft would **not** be a sufficient and appropriate security measure for.*

Threat Modeling document for Nahoft

This document describes the thought process behind the development of Nahoft as well as the risk model it was designed for. This document should be presented to and read by the users of Nahoft before use, —as well as in the readme file of the git repository of the app— as the misuse of the app could bear serious risks. It should be noted that Nahoft is a Proof of Concept application for specific use cases defined in this document.

Nahoft went through two rounds of penetration testing by Cure53, a respected German cybersecurity form founded by Dr. Mario Heidrich, and multiple security issues have been found, but in the conclusion of the second testing reports, the testers concluded that "**all discovered issues were remediated shortly after being reported to the Nahoft team. Therefore, the app now appears ready for secure use in production**."

As mentioned in the final pen-test report by Cure53 researchers, the threat model which the penetration testing engagement was mainly focused on, were "***threats arising from an attacker who manages to obtain the phone and control malicious, third-party Apps***".

This document outlines the threat models of users/situations who would benefit from using Nahoft, and threat models that Nahoft would not be a sufficient and appropriate security measure for.

Maintenance of the project

Developers of the app at United for Iran (U4I) and Operator Foundation (OF) are actively monitoring:

- Third party libraries used in Nahoft such as Sodium
- Nahoft's code on Github

In case of becoming aware of any critical vulnerabilities in the libraries, or being notified by security researches about Nahoft's code, that we concluded expose users to serious risk, they will:

- Apply the necessary fixes, release a new version in which those vulnerabilities are addressed on Google Play and on Nahoft's website
- Or if fixing the vulnerabilities deem infeasible, they will unlist Nahoft on Google Play, and notify users on main social media channels it was promoted or advertised, U4I's website, and Nahoft's Website. In that case, Nahoft's code on Github will be marked as archived and changed into a 'read-only' repository.

Also, if due to financial and operational constraints, U4I decided to stop maintaining the project, It will notify users on U4I's website, and Nahoft's Website, and Nahoft's code on Github will be marked as archived and changed into a 'read-only' repository.

Assumptions

Below is a list of developers' assumptions in designing and developing this app. This helps users to have a better understanding of the thought process behind the app and what are the scenarios in which they would be able to benefit from it.

- Usage within the context of Iran, in the situation of temporary or permanent internet shutdown
- Access to non-state-controlled government apps like Signal is limited or restricted
    - Because of difficulty and increased risk of communication in a situation of Internet shutdown, we recommend potential users to install, start using Nahoft, and familiarize their trusted network with it before such incidents. But conducting frequent daily communications with Nahoft when easy access to apps like Signal is still feasible does not seem necessary.
- Nahoft is run on an android device that isn't compromised already
- Adversary does not have physical access to user's unlocked phone
    - If the adversary gains physical access to user's unlocked phone, and user is forced to enter Nahoft's passcode under pressure from adversary, user is empowered to enter the destruction code instead to remove saved contacts and messages

What does Nahoft do and doesn't do (scope and limitations)

Here, we explain the narrowed down scenario in which Nahoft can protect users' communication and the limitations that users need to be aware of:

Obfuscation limitations

- It is unlikely to be detectable with any existing tools (developers tested it with the free tools available) as it does not use traditional, obsolete methods which those tools look for in order to detect hidden messages. For an adversary to detect messages encrypted by Nahoft, development of a new, custom tool would most likely be necessary. For the steganography feature, this would require an expert in forensic analysis of images. It would also likely be slow and thus costly to deploy at scale.
- Nahoft does not represent the most advanced obfuscation that can be produced, because it was not deemed to be necessary to defeat the adversary. We initially used only Persian letters. As this was deemed insufficient, we upgraded to Persian words.
- Despite points mentioned above, the authorities could still potentially develop an algorithm to know that messages are being encrypted using Nahoft.

Limitations in regard to end-point security

If an end-point device is already compromised and the adversary has gained persistence, benefits from using Nahoft may be limited or non-existent. Below are the example scenarios in which using Nahoft will not protect a user's desired confidentiality:

- Hardware backdoor: Hardware backdoors are backdoors in hardware, such as code inside hardware or firmware of computer chips. If the manufacturer of a device, has implemented such backdoors, and grants access to the adversary through that backdoor, the adversary might gain access to the messages before they are encrypted by Nahoft or after decryption with private key if the recipient's device is compromised. For this reason, we recommend users to install and use Nahoft on devices manufactured by Original Equipment Manufacturers (OEM) that are less likely to implement such backdoors and grant access to adversaries.
- Stalkerware: If an adversary has managed to install a stalkerware on a device that infiltrates keystrokes, it would gain access to the unencrypted version of messages. For example, **this can happen if a user's device is confiscated during an arrest operation, and then handed back to the user after their release**.

Who is this for/whom:

Who are the primary users of this app under what circumstances? Is it ready/recommended for public use? Is it recommended to use for sensitive conversations?

- If you're an Iranian android phone user in the situation of Internet shutdown, forced to use "National Internet Network" (NIN), using Nahoft would add an extra layer of confidentiality by using strong encryption algorithms.

Who is this NOT for

Nahoft does not provide reliable protection for users:

- Concerned about the revelation of their audience. Meta-data of communication is still accessible if you send the messages via the state-controlled messaging apps to communicate messages encrypted by Nahoft. For example if encrypted messages are communicated via Soroush Messenger, following information can be accessed:
    - Which Soroush users you have been messaging
    - In what frequency and at what times you have communicated with them
- Users who need to send audio and video
- Users with end-point devices that might have already been compromised (see "limitation in regard to end-point security)
- Users of unsecure third-party and custom keyboards: Nahoft does not protect you from malicious or otherwise unsecure third-party and custom keyboards as they have access to everything you type before it gets encrypted. Such keyboards with network access, can potentially communicate all typed characters to a remote server.

Potential risks associated with using Nahoft

Some risks are listed in above sections “Who is this NOT for” and “What does Nahoft do and doesn’t do (scope and limitations)”. Below, we listed some other potential risk scenarios:

- Arrest and coercion to enter the passcode
    - Mitigation: set up a destruction code and if pressured to enter the passcode, enter the destruction code instead
- Replica: Since Nahoft’s code is open-sourced and available on Github, adversaries might create replicas that similar user experience to the original app, but with weakened encryption
    - Mitigation: Download Nahoft only from trust-worthy sources such as its official Google Play page
- Screen Capture: Adversaries may use screen captures to collect information about applications running in the foreground and capture user data. “Applications running in the background can capture screenshots or videos of another application running in the foreground by using the Android MediaProjectionManager” [Att&ck].
    - Mitigation: Users should not grant consent for screen captures and should avoid enabling USB debugging
      (Important Note: **Screen Capture is disabled for Nahoft** but this is still something users need to be aware of. For example in case user decides to type the message in another app, and then copy it to Nahoft to encrypt)

Pentesting Results

- Link to round one. For the fix for each vulnerability, see commits with the same finding code and title in the code in Github.
- Link to round two. For the fix for each vulnerability, see commits with the same finding code and title in the code in Github.

Disclaimers / Transparency

- Sustainability of the project: see section “Maintenance of the project”
- Roadmap for future versions
    - U4I has currently no fixed timeline for releasing another version of Nahof, and is only committed to fixing critical security vulnerabilities
    - U4I is maintaining and updating a list of ideas for improvement in case funding for another version becomes available
    - For submitting vulnerability findings and improvement ideas add a comment on Nahoft’s github AND email Nahoft[at]united4iran.org.
- Contributions guideline: WIP

References

- First pen-test

- Second pen-test
- Addressing Findings of Cure53's Second Pen-test
- Github
- Google Play
- Nahoft
- Mitre Att&ck Screen Capture