ASSURED

SECURITY CONSULTANTS

Report

Installer Downloader Audit

Joachim Strömbergson, Johanna Abrahamsson

Project	Version	Date
MUL020	1	2025-03-20

REPORT

Project	Version	Date
MUL020	1	2025-03-20

Executive summary

Between 2025-03-10 and 2025-03-17 Assured Security Consultants performed a review of the distributed app download solution being developed by Mullvad VPN AB.

In scope for the audit was the installer downloader application, the script generating installer releases, and the installer metadata.

This report lists the security issues found, along with recommendations for fixing or mitigating them. In our conclusions we discuss the issues and address apparent patterns in areas where security is lacking.

Observations were made with the following risk severity assessments (number of issues):

Critical 0 High 0 Medium 0 Low 1 Note 3

Our recommendations can be summarized as follows:

- Review the integrity check in the script `4-make-release`
- Consider if the TOCTOU is really mitigated by the randomly generated directory name
- Increase the number of characters in the randomly generated directory name

Assured would like to thank Linus Färnstrand, David Lönnhager and Oskar Nyberg for their support during this audit. We are happy to answer any questions and provide further advice.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Constraints and disclaimer . . . 1
- 1.3 Project period and staffing . . . 1

2 Scope and methodology **2**
- 2.1 Scope . . . 2
- 2.2 Methodology . . . 2

3 Observations **3**
- 3.1 LOW Release script does not verify that the key that has signed the binary is trusted . . . 3
- 3.2 NOTE `deserialize_and_verify` function does not return the exact data that is signed and verified . . . 4
- 3.3 NOTE Short random directory name . . . 5
- 3.4 NOTE `thread_rng()` is deprecated in latest `rand` version . . . 5

4 Conclusions and recommendations **6**

1 Introduction

1.1 Background

Assured AB (Assured) was contracted by Mullvad VPN AB to perform an audit of a new app functionality that allows downloading of new versions of the Mullvad VPN app installer from CDN sources in a secure manner.

The solution is based on The Update Framework[1]. In short, the app retrieves metadata describing available versions of the installler from Mullvad. Based on the metadata, the app can determine if there is a new version of the installer available. If an update exists, the app downloads the new version from one of several sources.

The installer downloader functionality, the metadata structure, the metadata parser, the build script for installer releases as well as the tool which generates keys and the metadata are in scope for the audit.

1.2 Constraints and disclaimer

This report contains a summary of the observations made during the project period. This report should not be considered as a complete list of all vulnerabilities, security flaws and/or misconfigurations.

1.3 Project period and staffing

Assured started the project on 2025-03-10 and finished on 2025-03-17.

This report was last reviewed on 2025-03-20.

Involved in the audit were Assured consultants Joachim Strömbergson and Johanna Abrahamsson.

---

[1] `https://theupdateframework.io/`

2 Scope and methodology

2.1 Scope

In scope for the audit was the installer downloader function in the app including the metadata parser. The build script for installer releases was also in scope, along with the tool for generating signing keys and the metadata.

- The downloader-installer application function: `https://github.com/mullvad/mullvadvpn-app/tree/main/installer-downloader`
- The metadata parser function is part of the the mullvad update function: `https://github.com/mullvad/mullvadvpn-app/tree/main/mullvad-update`
- The build script: `https://github.com/mullvad/mullvadvpn-app/blob/main/installer-downloader/build.sh`
- The release scripts: `https://github.com/mullvad/mullvadvpn-app/blob/main/desktop/scripts/release/`
- The signing and metadata tool: `https://github.com/mullvad/mullvadvpn-app/tree/main/mullvad-update/meta`

2.2 Methodology

The audit has been performed based on manual code analysis and execution in combination with tools used to audit the code. The following tools were used:

- Cargo Audit
- Cargo Clippy
- Cargo Scan
- Emacs with LSP and tree-sitter for Rust

3 Observations

3.1 LOW Release script does not verify that the key that has signed the binary is trusted

The bash script that is used to generate and sign the metadata using the meta tool, `4-make-release`, downloads the build artifacts for the release before producing the metadata. The downloaded artifacts are checked against signatures which are fetched from the same server. However, the command that is used, `gpg2 --verify`, will return 0 if the signature matches the data in the file regardless of what key was used to produce the signature. A snippet of the script is included below for reference.

desktop/scripts/release/4-make-release

```
        echo ">>> Verifying integrity of $pkg_filename"
        if ! $gpg_cmd --verify "$pkg_path.asc" "$pkg_path"; then
            echo ""
            echo "!!! INTEGRITY CHECKING FAILED !!!"
            rm "$pkg_path" "$pkg_path.asc"
            exit 1
        fi
        echo ""
        echo "GOOD SIGNATURE FOR $pkg_filename"
        echo ""
```

We recommend looking into whether this is the intended behaviour or if the script should fail if the signature is not made with a trusted key. If this is the intended behaviour, we recommend clarifying this in the output of the script. If the intention is to verify the signature's authenticity we recommend using either the machine-parsable interface of gpg by supplying the options `--with-colons` and `--status-fd`, or using a tool specific to the purpose such as gpgv.

3.2 NOTE `deserialize_and_verify` function does not return the exact data that is signed and verified

The function `deserialize_and_verify` on the type `SignedResponse` implements deserialization and verification of signed `JSON` data. However, while the verification of the signature is performed on a canonicalized copy of the input data, the returned object contains the original input data. A snippet of the code is included below for reference.

While the data should remain unchanged through canonicalization, there is a risk that changes that does not affect the canonicalized form either by design or by implementation error may affect how the data is later processed. This risk can be eliminated by returning the canonicalized data instead.

mullvad-update/src/format/deserializer.rs

```
    // Serialize to canonical json format
    let canon_data = json_canon::to_vec(&partial_data.signed)
        .context("Failed to serialize to canonical JSON")?;

    // Check if the data is signed by our key
    key.0
        .verify_strict(&canon_data, &sig.0)
        .context("Signature verification failed")?;

    Ok(PartialSignedResponse {
        signatures: partial_data.signatures,
        signed: partial_data.signed,
    })
}
```

We recommend returning the canonicalized data, even if that may require an extra deserialization.

3.3 NOTE Short random directory name

For macOS, Mullvad has identified a possible Time-of-Check, Time-of-Use (TOCTOU) issue. The issue is caused by the downloader not running as a privileged user, but as the current user. The downloaded asset is therefore stored in a temporary directory. The asset could be modified between the digest for the asset being verified and the asset being used. But only by the current user.

The mitigation is to use a random name for the directory. The random directory name is generated in `temp.rs`. The generated random name consists of 10 alphanumeric characters. The entropy per alphanumeric character is about 6 bits. Given 10 characters, the total entropy for the random name is somewhat low, making guessing the directory not impossible.

We recommend to increase the length of the random name to 16 or more characters.

Of more concern is if the mitigation actually solves the TOCTOU. Would a malicious application running as a the current user be able to detect the creation of the temporary, randomly named directory, and perform the attack with some probability of success?

3.4 NOTE `thread_rng()` is deprecated in latest `rand` version

The downloader installer, and out of scope for the audit but observed during the audit, the Mullvad application at several places use the `thread_rng()` function from the to `rand` crate to get random numbers. The usage of the function in scope for the audit exists in `controller.rs` and `temp.rs`.

The version of the `rand` crate being used is `0.8.9`. In the latest version, `0.9.0`, the function `thread_rng()` has been deprecated and should be replaced with `rand::rng()` when switching to the new version.

4 Conclusions and recommendations

Based on our review of the source code, the new downloader installer solution seems to be well thought out and implemented.

Our observations has identified some minor issues and notes.

Our recommendations can be summarized as follows:

- Review the integrity check in the script `4-make-release`
- Consider if the TOCTOU is really mitigated by the randomly generated directory name
- Increase the number of characters in the randomly generated directory name