Nightly Labs

Security Assessment

January 21st, 2025 — Prepared by OtterSec

Bruno Halltari bruno@osec.io

Table of Contents

01 — Executive Summary

Overview

Nightly Labs engaged OtterSec to assess the `aptos-snap` program. This assessment was conducted between January 16th and January 17th, 2025. For more information on our auditing methodology, refer to Appendix B.

Key Findings

We produced 3 findings throughout this audit engagement.

In particular, we identified a vulnerability concerning the lack of transaction parameter visibility for users, which may expose them to scam transactions (OS-NLB-ADV-00). Additionally, the Snap module fails to handle newlines properly in dialog messages, allowing attackers to inject misleading text that may facilitate phishing or spoofing attacks (OS-NLB-ADV-01).

We also advised validating the URL format of networks to ensure that any URLs provided for network configuration are valid and correctly structured (OS-NLB-SUG-00).

02 — Scope

The source code was delivered to us in a Git repository at https://github.com/nightly-labs/aptos-snap. This audit was performed against commit 0b381a1.

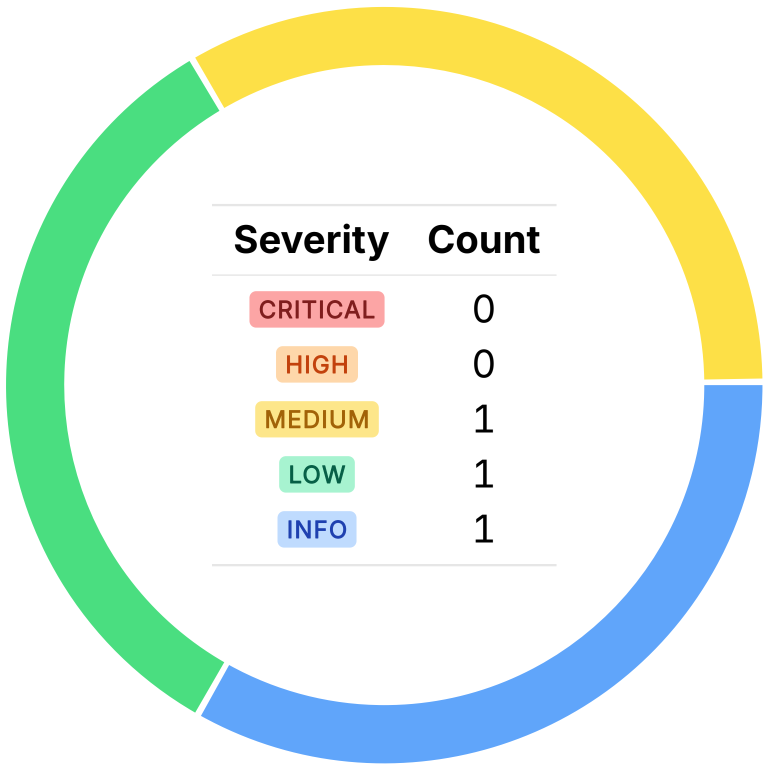
A brief description of the program is as follows:

Name	Description
aptos-snap	It enables secure interactions between dApps and a user’s Aptos account through MetaMask Snaps, facilitating actions such as account connection, network fetching or switching, transaction signing, and message signing—all accompanied by user confirmation dialogs.

03 — Findings

Overall, we reported 3 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings do not have an immediate impact but will aid in mitigating future vulnerabilities.



Severity	Count
CRITICAL	0
HIGH	0
MEDIUM	1
LOW	1
INFO	1

04 — Vulnerabilities

Here, we present a technical analysis of the vulnerabilities we identified during our audit. These vulnerabilities have *immediate* security implications, and we recommend remediation as soon as possible.

Rating criteria can be found in Appendix A.

ID	Severity	Status	Description
OS-NLB-ADV-00	MEDIUM	RESOLVED ✓	The lack of transaction parameter visibility for users may expose them to scam transactions.
OS-NLB-ADV-01	LOW	RESOLVED ✓	The Snap module fails to handle newlines properly in dialog messages, allowing attackers to inject misleading text that may facilitate phishing or spoofing attacks.

Lack of Transaction Transparency MEDIUM OS-NLB-ADV-00

Description

The current implementation allows the user to sign and submit transactions (`signAndSubmitTransaction`) without explicitly displaying or simulating transaction parameters. This lack of transparency creates a significant risk that a user might unknowingly authorize a malicious or fraudulent transaction, potentially compromising their account or funds.

>_ *packages/snap/src/index.tsx* RUST

```
export const onRpcRequest: OnRpcRequestHandler = async ({
  origin,
  request,
}) => {
    [...]
    case 'signAndSubmitTransaction': {
          const account = await getAccount();
          const result = await snap.request({
            method: 'snap_dialog',
            params: {
              type: 'confirmation',
              content: (
                <Box>
                  <Text>
                    <Bold>{origin}</Bold> wants to sign a transaction
                  </Text>
                  <Text>
                    using <Bold>{account.accountAddress.toString()}</Bold> account
                  </Text>
                  <Text>Confirm to sign this transaction.</Text>
                </Box>
              ),
            },
          });
          [...]
        }
    [...]
}
```

Remediation

Display the transaction parameters before confirmation, or simulate the transaction's execution and show a summary of its expected outcome to help users understand its effects.

Patch

Fixed in de5a944. Nightly is also commiting to introduce a full simulation.

Improper Input Sanitization LOW OS-NLB-ADV-01

Description

The Snap module allows interaction with users via confirmation dialogs for sensitive actions. However, improper handling of newline characters (`\n` or `\r`) in these dialogs may result in spoofing or phishing attacks. Newlines inject additional lines of text into the dialog box displayed to the user, creating a misleading appearance and tricking the user into believing the dialog contains genuine warnings or instructions. This may prompt users to take unsafe actions, such as approving transactions or granting access to their accounts.

Remediation

Strip all newline characters (`\n`, `\r`) from inputs displayed in dialogs.

Patch

Fixed in 7911177.

05 — General Findings

Here, we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent anti-patterns and may result in security issues in the future.

ID	Description
OS-NLB-SUG-00	Malformed `URLs` for network configurations may bypass verification, posing security risks.

Validation of Network URL Format

OS-NLB-SUG-00

Description

Validate the `URL` format of networks utilizing the new `URL()` constructor call to ensure that any `URLs` provided for network configuration are valid and correctly structured.

Remediation

Implement the above-mentioned suggestion.

A — Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings may be found in the General Findings.

CRITICAL

Vulnerabilities that immediately result in a loss of user funds with minimal preconditions.

Examples:

- Misconfigured authority or access control validation.
- Improperly designed economic incentives leading to loss of funds.

HIGH

Vulnerabilities that may result in a loss of user funds but are potentially difficult to exploit.

Examples:

- Loss of funds requiring specific victim interactions.
- Exploitation involving high capital requirement with respect to payout.

MEDIUM

Vulnerabilities that may result in denial of service scenarios or degraded usability.

Examples:

- Computational limit exhaustion through malicious input.
- Forced exceptions in the normal user flow.

LOW

Low probability vulnerabilities, which are still exploitable but require extenuating circumstances or undue risk.

Examples:

- Oracle manipulation with large capital requirements and multiple transactions.

INFO

Best practices to mitigate future security risks. These are classified as general findings.

Examples:

- Explicit assertion of critical internal invariants.
- Improved input validation.

B — Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

When auditing the design of a program, we aim to ensure that the overall economic architecture is sound in the context of an on-chain program. In other words, there is no way to steal funds or deny service, ignoring any chain-specific quirks. This usually requires a deep understanding of the program's internal interactions, potential game theory implications, and general on-chain execution primitives.

One example of a design vulnerability would be an on-chain oracle that could be manipulated by flash loans or large deposits. Such a design would generally be unsound regardless of which chain the oracle is deployed on.

On the other hand, auditing the program's implementation requires a deep understanding of the chain's execution model. While this varies from chain to chain, some common implementation vulnerabilities include reentrancy, account ownership issues, arithmetic overflows, and rounding bugs.

As a general rule of thumb, implementation vulnerabilities tend to be more "checklist" style. In contrast, design vulnerabilities require a strong understanding of the underlying system and the various interactions: both with the user and cross-program.

As we approach any new target, we strive to comprehensively understand the program first. In our audits, we always approach targets with a team of auditors. This allows us to share thoughts and collaborate, picking up on details that others may have missed.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.