Veridise. Auditing Report

Hardening Blockchain Security with Formal Methods

FOR

Partisia Blockchain Metamask Snap

Veridise Inc.

June 25, 2023

I Prepared For:

Partisia

I Prepared By:

Benjamin Mariano
Nicholas Brown

I Contact Us: contact@veridise.com

I Version History:

June 25, 2023 V1

Contents

1 Executive Summary

From June 14, 2023 to June 16, 2023, Partisia engaged Veridise to review the security of the Partisia Blockchain Metamask Snap, a MetaMask snap application that allows users to retrieve Partisia Blockchain addresses and sign transactions for the Partisia Blockchain. The auditing strategy involved extensive manual analysis/auditing of the source code performed by Veridise engineers.

Code assessment. The Partisia developers provided the source code of the Partisia Blockchain Metamask Snap project for review. The code includes a number of tests that were useful for auditors to better understand the code. The code does not currently have significant external documentation; however, the codebase is small, consisting of less than 200 lines of total code, including significant comments within the code itself.

Summary of issues detected. The audit uncovered 3 total issues. These issues included a lack of transaction payload validation (V-PAR-VUL-001), unnecessary private key use (V-PAR-VUL-002), and an unclear user message that could be vulnerable to social engineering attacks (V-PAR-VUL-003).

Disclaimer. We hope that this report is informative but provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall Veridise or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.

2 Project Dashboard

Table 2.1: Application Summary.

Name	Version	Type	Platform
Partisia Metamask Snap	487032cc	Typescript	MetaMask

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
June 14 - June 16, 2023	Manual	2	2 person-weeks

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	0	0
Low-Severity Issues	0	0
Warning-Severity Issues	1	1
Informational-Severity Issues	2	2
TOTAL	3	3

Table 2.4: Category Breakdown.

Name	Number
Data Validation	1
Logic Error	1
Usability Issue	1

3 Audit Goals and Scope

3.1 Audit Goals

The engagement was scoped to provide a security assessment of Partisia's source code.

In our audit, we sought to answer the following questions:

- Can a user's private key be stolen?
- Can a malformed transaction be signed?
- Are best practices around private key usage followed (e.g., are private keys only requested when necessary? Are private keys ever exposed to public channels? etc.)
- Is a transaction properly signed according to the expectations of the blockchain?
- Is the correct address retrieved on a address retrieval request?

3.2 Audit Methodology & Scope

Audit Methodology. To address the questions above, our audit involved a detailed manual analysis of the code by human experts.

Scope. The scope of this audit is limited to the source code of the Snap in the `snap/packages/snap/src` folder.

Methodology. Veridise auditors inspected provided tests, and read the Partisia Blockchain Metamask Snap documentation. They then began a manual audit of the code.

3.3 Classification of Vulnerabilities

When Veridise auditors discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise. Table 3.1 shows how our auditors weigh this information to estimate the severity of a given issue.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

In this case, we judge the likelihood of a vulnerability as follows in Table 3.2:

In addition, we judge the impact of a vulnerability as follows in Table 3.3:

Table 3.2: Likelihood Breakdown

Not Likely	A small set of users must make a specific mistake
Likely	Requires a complex series of steps by almost any user(s) - OR - Requires a small set of users to perform an action
Very Likely	Can be easily performed by almost anyone

Table 3.3: Impact Breakdown

Somewhat Bad	Inconveniences a small number of users and can be fixed by the user
Bad	Affects a large number of people and can be fixed by the user - OR - Affects a very small number of people and requires aid to fix
Very Bad	Affects a large number of people and requires aid to fix - OR - Disrupts the intended behavior of the protocol for a small group of users through no fault of their own
Protocol Breaking	Disrupts the intended behavior of the protocol for a large group of users through no fault of their own

4 Vulnerability Report

In this section, we describe the vulnerabilities found during our audit. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status (i.e., acknowleged, fixed, etc.). Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-PAR-VUL-001	Lack of transaction payload validation	Warning	Acknowledged
V-PAR-VUL-002	Unnecessary use of private key	Info	Acknowledged
V-PAR-VUL-003	Opaque user confirmation message	Info	Acknowledged

4.1 Detailed Description of Issues

4.1.1 V-PAR-VUL-001: Lack of transaction payload validation

Severity	Warning	**Commit**	487032c
Type	Data Validation	**Status**	Acknowledged
File(s)	`index.ts`		
Location(s)	onRpcRequest()		
Fix Commit	N/A		

The `sign_transaction` RPC request only validates the length of the payload to ensure that the necessary header info is present. There is no validation of the rest of the transaction payload. This can result in the user signing malformed transactions which may fail or result in unexpected behavior.

Impact A malformed transaction will be presented to the user the same way as any other transaction, so if the transaction doesn't fail when sent to the blockchain, unexpected behavior can occur. Depending on how these transactions are handled, this could cause problems.

Recommendation Validate transaction content before sending transaction to the blockchain, unless any transaction with invalid parameters will revert when being executed on the blockchain.

Developer Response At this time, additional transaction validation would require expensive and/or complicated logic to retrieve additional information. Therefore, no additional validation will be added into the current version. However, the plan is to add more validation on future iterations of the snap.

4.1.2 V-PAR-VUL-002: Unnecessary use of private key

Severity	Info	**Commit**	487032c
Type	Logic Error	**Status**	Acknowledged
File(s)	`index.ts`		
Location(s)	onRpcRequest()		
Fix Commit	N/A		

The `get_address` request retrieves an account address from a private key. It does this by getting the private key from the user, recovering the associated public key, taking the hash of this public key, and retrieving the first 24 bytes of that hash (prepended with `00`). As the private key is only used to retrieve the public key, it is safer to only request the public key.

Impact The private key of a user is sensitive and any use creates the potential risk of it being stolen. Therefore, the private key should only be requested when necessary.

Recommendation Request the public key instead of the private key for retrieving the address. This should be possible by using the `snap_getBip32PublicKey` function.

```
const publicKey = await snap.request({
  method: 'snap_getBip32PublicKey',
  params: {
    // The path and curve must be specified in the initial permissions.
    path: ['m', "44'", "3757'", "0'", '0', '0'],
    curve: 'secp256k1',
    compressed: false,
  },
});
```

Developer Response On experimentation with the suggested API, it does not appear to function as expected in this context. To avoid potential errors, the developers have opted to keep the current API.