Veridise. Auditing Report

Hardening Blockchain Security with Formal Methods

FOR

cubist™

TypeScript SDK
MetaMask Snap

Veridise Inc.
September 9, 2023

► **Prepared For:**

Cubist
https://cubist.dev/

► **Prepared By:**

Benjamin Mariano
Alberto Gonzalez
Xiangan He

► **Contact Us:** contact@veridise.com

► **Version History:**

July 31, 2023	V1
Aug. 30, 2023	V2

Contents

Contents iii

1 Executive Summary 1

2 Project Dashboard 3

3 Audit Goals and Scope 5

3.1 Audit Goals 5

3.2 Audit Methodology & Scope 5

3.3 Classification of Vulnerabilities 5

4 Vulnerability Report 7

4.1 Detailed Description of Issues 8

4.1.1 V-CUB-VUL-001: Unsuccessfully revoked tokens can be reused 8

4.1.2 V-CUB-VUL-002: Incorrect assumption about chain id 9

4.1.3 V-CUB-VUL-003: setName function does not set anything 10

4.1.4 V-CUB-VUL-004: API misuse with disallowed chain ID 11

4.1.5 V-CUB-VUL-005: Non-null assertions might cause run-time errors . . . 12

4.1.6 V-CUB-VUL-006: Users cannot be removed from role 13

4.1.7 V-CUB-VUL-007: Duplicated error handling 14

4.1.8 V-CUB-VUL-008: Incorrect comments 15

4.1.9 V-CUB-VUL-009: No Unstake / Stake Operations in Request Handler . . 16

4.1.10 V-CUB-VUL-010: Error handling on login exposes user secrets 17

4.1.11 V-CUB-VUL-011: Fix token docstrings to align with naming 18

4.1.12 V-CUB-VUL-012: Confusing variable use 19

Executive Summary 1

From July 19, 2023 to July 27, 2023, Cubist engaged Veridise to review the security of their Cubist MetaMask Snap and TypeScript SDK. The review covered Cubist's new MetaMask snap which is designed to allow users to easily interact with their transaction signing APIs. Veridise conducted the assessment over 15 person-days, with 3 engineers reviewing code over 5 days from commits `951a820` - `031312f`. The auditing strategy involved a extensive manual auditing by Veridise engineers.

Code assessment. The Cubist MetaMask Snap and TypeScript SDK developers provided the source code of the Cubist MetaMask Snap and TypeScript SDK contracts for review. The code is fairly well tested, with tests of most expected behaviors as well as unit tests of internal functions. Documentation is somewhat limited, although the code is fairly well documented via descriptive comments within the source files.

Summary of issues detected. The audit uncovered 12 issues. The most severe error was a medium-level issue where a key could silently remain unrevoked after users expected it to be revoked (V-CUB-VUL-001). Veridise auditors also reported a number of other minor (info and warning-level) issues, including incomplete functions (V-CUB-VUL-003) and missing functionality (V-CUB-VUL-006, V-CUB-VUL-009).

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.

Project Dashboard 2

Table 2.1: Application Summary.

Name	Version	Type	Platform
Snap and TypeScript SDK	`951a820` - `031312f`	TypeScript	MetaMask

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
July 19 - July 27, 2023	Manual	3	15 person-days

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	1	1
Low-Severity Issues	0	0
Warning-Severity Issues	5	5
Informational-Severity Issues	6	6
TOTAL	12	12

Table 2.4: Category Breakdown.

Name	Number
Usability Issue	4
Logic Error	3
Maintainability	2
Authorization	1
Documentation	1
Information Leakage	1

3 Audit Goals and Scope

3.1 Audit Goals

The engagement was scoped to provide a security assessment of Cubist MetaMask Snap and TypeScript SDK's smart contracts. In our audit, we sought to answer the following questions:

- Can secret information of a user be leaked to any publicly accessible channels?
- Is secret information stored securely for maintaining active sessions?
- Are users prompted before risky or potentially dangerous actions are undertaken (e.g., signing transactions)?
- Do user-facing APIs behave as expected?
- Does the Snap leave users exposed to potential phishing scams?
- Are the OpenAPI specifications respected by all calls made?

3.2 Audit Methodology & Scope

Audit Methodology. To address the questions above, our audit involved a manual review of the code by Veridise's engineers.

Scope. The scope of this audit is limited to the `packages/` folder of the source code provided by the Cubist MetaMask Snap and TypeScript SDK developers, which contains the smart contract implementation of the Cubist MetaMask Snap and TypeScript SDK, including both the `snap/` directory which contains the Snap implementation as well as the `sdk/` folder, which contains methods for interacting with the Cubist API.

Methodology. Veridise auditors reviewed the reports of previous audits for Cubist, inspected the provided tests, and read the Cubist MetaMask Snap and TypeScript SDK documentation. They then began a manual audit of the code. During the audit, the Veridise auditors met with the Cubist MetaMask Snap and TypeScript SDK developers to ask questions about 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:

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

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

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

Vulnerability Report 4

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., acknowledged, fixed, etc.). Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-CUB-VUL-001	Unsuccessfully revoked tokens can be reused	Medium	Fixed
V-CUB-VUL-002	Incorrect assumption about chain id	Warning	Acknowledged
V-CUB-VUL-003	setName function does not set anything	Warning	Fixed
V-CUB-VUL-004	API misuse with disallowed chain ID	Warning	Fixed
V-CUB-VUL-005	Non-null assertions might cause run-time errors	Warning	Acknowledged
V-CUB-VUL-006	Users cannot be removed from role	Warning	Acknowledged
V-CUB-VUL-007	Duplicated error handling	Info	Fixed
V-CUB-VUL-008	Incorrect comments	Info	Fixed
V-CUB-VUL-009	No Unstake / Stake Operations in Request Handle	Info	Acknowledged
V-CUB-VUL-010	Error handling on login exposes user secrets	Info	Fixed
V-CUB-VUL-011	Fix token docstrings to align with naming	Info	Fixed
V-CUB-VUL-012	Confusing variable use	Info	Fixed

4.1 Detailed Description of Issues

4.1.1 V-CUB-VUL-001: Unsuccessfully revoked tokens can be reused

Severity	Medium	**Commit**	01aca78
Type	Authorization	**Status**	Fixed
File(s)	`session.ts`		
Location(s)	handleLogout()		
Confirmed Fix At	a0aee44		

The `handleLogout` function appears to clear the state regardless of whether the `session.revoke()` call was successful or not. If `session.revoke()` fails (for example, because the user doesn't have a management token, or due to any other error), the application still proceeds to clear the state, which might mislead the user into believing that their API token has been successfully revoked.

Impact This introduces a security risk because a user no longer believes their token is valid, so may reveal it without knowing the risks. If the token falls into the wrong hands, it could be misused by a malicious actor.

Recommendation A more reliable approach might be to only clear the state if the `session.revoke()` call was successful. If it fails, the function could return an error to the user or throw an exception on logout.

Developer Response The developers have opted for a solution that still completes the logout with a session that was not revoked, but communicates this information clearly to the user so there is no confusion and instructs them on how to use the CLI to revoke their token. They believe this approach should be fine as their tokens are typically short-lived and so will expire relatively quickly even without a specific revocation.

4.1.2 V-CUB-VUL-002: Incorrect assumption about chain id

Severity	Warning	**Commit**	01aca78
Type	Usability Issue	**Status**	Acknowledged
File(s)	`sdk/src/schema.ts`		
Location(s)	N/A		
Confirmed Fix At	N/A		

The chain_id field of the EthSignRequest object assumes that it fits in 64 bits. Per the discussion here, chain ids may be greater due to using hash functions to compute them.

```
Eth1SignRequest: {
  /**
    * Format: int64
    * @description Chain id to set in the given transaction.
    */
  chain_id: number;
  /** @description ‘TypedTransaction‘ object as an untyped JSON value. */
  tx: Record<string, never>;
};
```

Impact The SDK will not support EVM blockchains with chain ids that use more than 64 bits.

Recommendation Define `chain_id` as a `BigInt`

Developer Response At this time, the developers will continue to use the interface as is because they rely on ethers-rs which uses a `u64`, so changing the interface would require a large change.

4.1.3 V-CUB-VUL-003: setName function does not set anything

Severity	Warning	**Commit**	01aca78
Type	Logic Error	**Status**	Fixed
File(s)	`sdk/src/org.ts`		
Location(s)	setName()		
Confirmed Fix At	32594d0		

The `setName` function only validates the syntactic form of the `name` argument but does not change any state.

```
/** Set the human-readable name for the org.
   * @param {string} name The new human-readable name for the org (must be
    alphanumeric).
   * @example my_org_name
   * */
  async setName(name: string) {
    // @audit Check this.
    if (!/^[a-zA-Z0-9_]{3,30}$/.test(name)) {
      throw new Error("Org name must be alphanumeric and between 3 and 30 characters
    ");
    }
  }
```

Impact The `name()` function will always return `null`, although `setName` was called with a valid name.

Recommendation Update the function by making a patch request to update the organization to change its name. From `schema.ts`:

```
"/v0/org/{org_id}": {
    /**
     * Get Org
     * @description Get Org
     *
     * Retrieves information about an organization.
     */
    get: operations["getOrg"];
    /**
     * Update Org
     * @description Update Org
     *
     * Update organization attributes (enabled flag, name, and policies).
     */
    patch: operations["updateOrg"];
};
```

Developer Response Suggested fix implemented in commit `32594d0`.

4.1.4 V-CUB-VUL-004: API misuse with disallowed chain ID

Severity	Warning	**Commit**	01aca78
Type	Documentation	**Status**	Fixed
File(s)	`sdk/src/key.ts`		
Location(s)	createKeys()		
Confirmed Fix At	28e77af		

The function `createKeys` always sets the `chain_id` value to `0` before making a call to create keys. However, the OpenAPI spec shown below expects a chain ID with a minimum value of `1.0`.

```
"CreateKeyRequest": {
        "type": "object",
        "required": ["chain_id", "key_type", "count"],
        "properties": {
          "chain_id": {
            "type": "integer",
            "format": "int64",
            "description": "Chain id for which the key is allowed to sign messages",
            "example": 5,
            "minimum": 1.0
          },
          ...
      }
```

Impact This could lead to all key creation requests being rejected if the `chain_id` is determined to be invalid.

Recommendation Use a valid chain ID or remove the unused parameter if possible.

Developer Response The minimum value has been updated to include 0.

4.1.5 V-CUB-VUL-005: Non-null assertions might cause run-time errors

Severity	Warning	**Commit**	01aca78
Type	Usability Issue	**Status**	Acknowledged
File(s)	sdk/src/token.ts		
Location(s)	create()		
Confirmed Fix At	N/A		

The `create` function in the `token.ts` file make use of non-null assertion `!` in `data.session_info`:

```
return new Token(cs.withSignerToken(data.token), <SignerSessionObject>{
      org_id: orgId,
      role_id: roleId,
      purpose: req.purpose,
      token: data.token,
      // @audit Best practice to not use non-null assertions, may cause runtime
    errors.
      session_info: data.session_info!,
 });
```

Actually, the `CreateTokenResponse` might return `null` as `session_info` :

```
CreateTokenResponse: {
  content: {
    "application/json": {
        session_info?: components["schemas"]["ClientSessionInfo"] | null;
        /**
          * @description Token to be used for signing auth. Requests to signing
    endpoints
          * should include this value in the ‘Authorization‘ header
          */
        token: string;
      };
    };
};
```

Impact If the assumption that `data.session_info` is never going to be `null` happens to be incorrect, then an error will occur and disrupt the execution of the code.

Recommendation There are a couple of patterns recommended instead of non-null assertions. Using any of these should avoid the problem.

Developer Response Developers indicated this assumption will always hold and therefore the fix here is to update the documentation.

4.1.6 V-CUB-VUL-006: Users cannot be removed from role

Severity	Warning	**Commit**	01aca78
Type	Usability Issue	**Status**	Acknowledged
File(s)	`sdk/src/role.ts`		
Location(s)	N/A		
Confirmed Fix At	N/A		

For roles, a user can be added via `addRole` but cannot be removed

Impact A user cannot be removed from a role

Recommendation Add a function for removing a user from a role if this is desired behavior.

Developer Response For now, the developers have decided not to support this operation. They may opt to add this functionality at a later date.

4.1.7 V-CUB-VUL-007: Duplicated error handling

Severity	Info	**Commit**	01aca78
Type	Logic Error	**Status**	Fixed
File(s)	`sdk/src/index.ts`		
Location(s)	aboutMe(), getOrg()		
Confirmed Fix At	32594d0		

The functions `aboutMe` and `getOrg` both check for errors directly in their code, rather than relying on `assertOk` from `sdk/src/env.ts` which is used in other locations to handle error checking.

Impact This could lead to unexpected inconsistencies in error handling, especially if the implementation of error handling in `assertOk` is changed in the future.

Recommendation For consistency and to avoid any errors due to inconsistency, developers should use the `assertOk` function here as is used throughout the rest of the code.

Developer Response Suggested fix implemented in commit `32594d0`.

4.1.8 V-CUB-VUL-008: Incorrect comments

Severity	Info	**Commit**	01aca78
Type	Usability Issue	**Status**	Fixed
File(s)	`sdk/src/org.ts`		
Location(s)	id(), fetch()		
Confirmed Fix At	32594d0		

The functions `id` and `fetch` both have inaccurate comments that appear to have been copied from other functions

Impact Inaccurate comments can lead to developer and user errors when intended behavior is not properly understood.

Recommendation Fix comments.

Developer Response Fixed in commit `32594d0`.

4.1.9 V-CUB-VUL-009: No Unstake / Stake Operations in Request Handler

Severity	Info	**Commit**	01aca78
Type	Logic Error	**Status**	Acknowledged
File(s)	`index.ts`		
Location(s)	onRpcRequest()		
Confirmed Fix At	32594d0		

`onRpcRequest` is a handler for receiving signing requests of multiple sorts.

Even though the ETH staking operations are included in the Cubist SDK, it is not handled in the RPC request handler.

Impact Staking and unstaking is not supported through the snap.

Recommendation Implement logic to handle staking/unstaking requests

Developer Response This requires some discussion among the team as to the desired behavior. For now, they have opted to remove the `eth2_sign` capability from the Snap and will revisit at a later time.

4.1.10 V-CUB-VUL-010: Error handling on login exposes user secrets

Severity	Info	**Commit**	01aca78
Type	Information Leakage	**Status**	Fixed
File(s)	`session.ts`		
Location(s)	handleLogin()		
Confirmed Fix At	a0aee44		

The current implementation of the function `handleLogin` includes a block that catches exceptions, and in the event of an error, it displays a message to the user that includes the error details.

When processing sensitive data like a base64-encoded JSON API token, the error can reveal the token to the user. If the parsing fails due to a syntax error, function throws an error with `JSON.parse` that includes a segment of the input string, in this case, the token.

Impact This issue is indicated as `info`-level severity as it is not clear that this issue can be easily exploited — information in snap dialogues should be restricted to the MetaMask extension (assuming the whole browser has not been compromised). The main concern here is that a user may copy and paste their error to attempt to find a solution (and could even post it) and unknowingly could reveal a secret. In general, showing secret information in plaintext is considered risky and should only be done in cases when no other option is available.

Recommendation At this time, Veridise auditors were unable to find a way that any information within a snap dialog, whether it be within the prompt or plaintext, can be leaked to other browser processes or snaps. However, despite this, we still suggest changing the behavior because (1) printing out an error code could be almost as informative with no risk and (2) a naive user could copy and paste the error message (including their secret information) and share it unknowingly (especially if the error contains their secret information in an encoded format).

Developer Response The developers removed the printing of the error message.

4.1.11 V-CUB-VUL-011: Fix token docstrings to align with naming

Severity	Info	**Commit**	01aca78
Type	Maintainability	**Status**	Fixed
File(s)	`signer_session.ts`		
Location(s)	N/A		
Confirmed Fix At	32594d0		

Recently, the developers seem to have refactored code by renaming `Token` related information to `SignerSession`. Documentation relating to `Token` info, however, are still in multiple places where `SignerSession` code stands.

Impact Future contributors may be confused by the lack of consistency between the purpose of `Tokens` and `SignerSession`.

Recommendation For consistency and maintainability, those docstrings should be adjusted to match with the refactor.

Developer Response Fixed in commit `32594d0`.

4.1.12 V-CUB-VUL-012: Confusing variable use

Severity	Info	**Commit**	01aca78
Type	Maintainability	**Status**	Fixed
File(s)	`snap/src/types.ts`		
Location(s)	santize()		
Confirmed Fix At	32594d0		

In the function `sanitize` the following code is used to recursively check that a given object conforms to the expected shape:

```
for (const [key, type] of Object.entries(shape)) {
    ...

    if (typeof shape[key] === "object") {
      // validate inner object
      obj[key] = sanitize(val, shape[key] as Shape);
    } else {
      ...
    }
  }
```

In multiple places, `shape[key]` is used, even though this value is identical to `type` as defined in the for loop header.

Impact This could confuse future developers as it subtly implies a difference between `shape[key]` and `type`.

Recommendation For consistency and maintainability, the same variable name should be used everywhere.

Developer Response Fixed in commit `32594d0`.