HACKEN

Snap Code Review And Security Analysis Report

Customer: Multiverse X

Date: 29/05/2024

We express our gratitude to the Multiverse X team for the collaborative engagement that enabled the execution of this Security Assessment.

MultiversX is a highly scalable, decentralized blockchain network designed for next-generation applications. It leverages adaptive state sharding and a secure proof-of-stake consensus mechanism to provide an efficient, scalable, and secure blockchain platform.

Language: TypeScript, JavaScript

Tags: [Snap]

Timeline: 14/05/2024 - 17/05/2024

Review Scope

Repository	https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/
Commit	992c22e

Audit Summary

10/10	n/a	n/a	n/a
Security score	Code quality score	Test coverage	Documentation quality score

Total 10/10

The system users should acknowledge all the risks summed up in the risks section of the report

4	4	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	1
Low	1

Vulnerability	Status
F-2024-2697 - Dependency Vulnerabilities	Fixed
F-2024-2701 - Potential for Exposing Sensitive Data	Fixed
F-2024-2702 - Insecure Compiler Flags	Fixed
F-2024-2832 - Insecure Handling of Private Keys	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Snap Code Review and Security Analysis Report for Multiverse X
Audited By	Stephen Ajayi
Approved By	Stephen Ajayi
Website	https://multiversx.com
Changelog	17/05/2024 - Preliminary Report

Table of Contents

System Overview

MultiversX, previously known as Elrond, is a highly scalable, decentralized blockchain network designed for next-generation applications. It leverages adaptive state sharding and a secure proof-of-stake consensus mechanism to provide an efficient, scalable, and secure blockchain platform. MultiversX is built to support a wide variety of blockchain protocols beyond Ethereum, enabling robust and versatile decentralized applications (dApps).

Audit Focus: MetaMask Snap

The audit conducted on the MultiversX MetaMask Snap focused on the permissions and security of the Snap's functionalities.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's Snap project.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **1** medium, and **1** low severity issues. Out of these, **4** issues have been addressed and resolved, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's Snap yields an overall score of **10**. This score reflects the combined evaluation of the security aspects of the project.

Findings

Vulnerability Details

F-2024-2832 - Insecure Handling of Private Keys - Medium

Description: The function `getAddress` uses `getWalletKeys` to retrieve the user's public key. The underlying `getWalletKeys` function potentially exposes private keys which can compromise user security by allowing unauthorized access to cryptographic material.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Classification

CVSS 4.0: 6.3 (/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:L/SC:N/SI:N/SA:L)

Known Vulnerability (CVE):

- CWE-934 | OWASP Top Ten 2013 Category A6 - Sensitive Data Exposure

Severity: Medium

Recommendations

Remediation:

- Refactor any function that currently exposes or could expose private keys to only handle or return public keys or addresses.
- Modify all cryptographic functions to never output private key material. Use secure scopes and environments for handling private keys.

Sample Fix:

Modify `getWalletKeys` to ensure no private keys are returned or exposed through the interface.

```
// Updated getWalletKeys function to never expose private keys.
export const getWalletKeys = async () => {
// Existing secure implementation details...
const publicKey = userSecret.generatePublicKey().toAddress().bech32(
);
```

```
return { publicKey };
};
```

Resolution: The `getWalletKeys` was modified to ensure no private keys were exposed

```
import { KeyOps } from "./operations/KeyOps";

/**
* This wallet uses a single account/address.
*/
export const getAddress = async (): Promise<string> => {
const keyOps = new KeyOps();
return await keyOps.getPublicKey();
};
```

Evidences

Proof of Concept (PoC):

Location: src/getAddress.ts, src/private-key.ts

Reproduce: The function `getWalletKeys` returns an object containing the `privateKey` directly:

```
export const getAddress = async (): Promise<string> => {
const { publicKey } = await getWalletKeys();
return publicKey;
};
```

Also verify that the `getWalletKeys` function should exposes sensitive cryptographic keys by returning private keys within the function's output.

```
return {
privateKey: node.privateKeyBytes,
publicKey: userSecret.generatePublicKey().toAddress().bech32(),
userSecret: userSecret,
};
```

F-2024-2702 - Insecure Compiler Flags - Low

Description:

The setting `"noImplicitAny": false` in the TypeScript compiler options allows implicit 'any' types. This reduces the type-safety of the code, potentially leading to runtime errors that are not caught at compile time, which could be exploited if they lead to unpredictable application behavior.

The compiler option `"sourceMap": true` is enabled, which can lead to the generation and exposure of source map files. These files can inadvertently reveal source code logic and structure in a production environment, aiding attackers in understanding and exploiting the application.

The `"skipLibCheck": true` option bypasses type checking of declaration files `(.d.ts)`. While this can improve compilation times, it also means that incorrect or malicious type declarations in third-party libraries could go unchecked.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Classification

CVSS 4.0: 2.3 (/AV:N/AC:H/AT:N/PR:L/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N)

Severity: Low

Recommendations

Remediation:

- Review and enable TypeScript's strict mode options, such as `strictNullChecks, strictBindCallApply,` and `strictFunctionTypes`, to ensure robust type-checking and reduce runtime errors.
- Configure build pipelines to handle different environments (development vs. production) appropriately. Exclude source maps and any non-essential files from production deployments

Sample Fix:

Enable strict type checks by setting `"noImplicitAny": true` to enforce type safety throughout the codebase.

```
"noImplicitAny": true,
```

Ensure that source maps are not included in production builds. This can be controlled by conditional configurations or build scripts that disable source map generation for production.

```
"sourceMap": process.env.NODE_ENV === 'production' ? false : true,
```

Consider disabling this option, especially in larger projects where reliability and security are paramount, to ensure all library types are validated.

```
"skipLibCheck": false,
```

Resolution:

```
"noImplicitAny": true,
"sourceMap": false,
"skipLibCheck": false
```

Evidences

Proof of Concept (PoC):

Location: tsconfig.json

Reproduce:

```
"noImplicitAny": false,
"sourceMap": true,
"skipLibCheck": true,
```

F-2024-2697 - Dependency Vulnerabilities - Info

Description: Vulnerabilities in dependencies can expose your application to security risks. The versions specified in `dependencies` and `devDependencies` might contain known vulnerabilities that have been fixed in later releases

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Classification

Severity: Info

Recommendations

Remediation:

- Automate dependency updates using tools like Dependabot or Renovate.
- Regularly run `npm audit` or similar tools to check for vulnerabilities and apply patches or updates promptly.

Sample Fix:

Utilize tools like `npm audit` to identify and mitigate vulnerabilities in the dependencies.

```
npm update
npm audit fix
```

Resolution:

```
"dependencies": {
"@multiversx/sdk-core": "13.1.0",
"protobufjs": "7.3.0"
},
```

Evidences

Proof of Concept (PoC):

Location: package.json

Reproduce: Dependencies like `"@multiversx/sdk-core": "^12.18.0"` and others might have known vulnerabilities in the used versions.

```
npm audit
# npm audit report

protobufjs 7.0.0 - 7.2.4
Severity: critical
protobufjs Prototype Pollution vulnerability - https://github.com/ad
visories/GHSA-h755-8qp9-cq85
fix available via `npm audit fix --force`
Will install @multiversx/sdk-core@13.1.0, which is a breaking change
node_modules/protobufjs
@multiversx/sdk-core 12.5.0 - 13.0.0-beta.18
Depends on vulnerable versions of protobufjs
node_modules/@multiversx/sdk-core

2 critical severity vulnerabilities
```

F-2024-2701 - Potential for Exposing Sensitive Data - Info

Description: Update the `snap.manifest.json` to include a comprehensive, clear description of why the `snap_getBip32Entropy` permission is necessary, detailing the risks and benefits. This ensures transparency and helps users make informed decisions.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Classification

Severity: Info

Recommendations

Remediation: Each time the Snap attempts to access this permission, prompt the user with a detailed consent form that must be actively approved. This form should clarify what the action involves and the potential risks.

Sample Fix:

Enhance the `snap.manifest.json` to include detailed descriptions of each permission. This should include why each permission is needed and the security.

```
"permissions": {
"snap_getBip32Entropy": {
"description": "This permission allows MultiversX Snap to manage acc
ounts and assets on various blockchain networks by deriving keys fro
m your secret recovery phrase. You Secrets are safe and not revealed
."
{
"path": [
.................
]
}"
```

Evidences

Proof of Concept (PoC):

Location: snap.manifest.json

Reproduce:

As stated in the manifest, the permission allows access to a specific `BIP32` derivation path which might be linked to wallet seed generation.

```
"snap_getBip32Entropy": [
{
"path": [
"m",
"44'",
"508'",
"0'",
"0'",
"0'"
],
"curve": "ed25519"
}
]
},
```

Observation Details

F-2024-2830 - Floating Point Precision and Rounding Errors - Info

Description: The function `format` takes a large numerical string (`big`), a denomination (`denomination`), and a number of decimal places (`decimals`) to format the number into a human-readable form with a specific precision. Due to the manipulation of numeric values as strings and the method of inserting a decimal point, there is a potential for precision and rounding errors. These can result in incorrect calculations or representations, which is particularly problematic in financial applications where accuracy is crucial.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Recommendations

Remediation:

- **Use Libraries for Decimal Calculations:** Replace custom string manipulation for numeric operations with libraries like `bignumber.js` or `decimal.js` to handle numbers safely and more accurately.
- **Avoid Reinventing the Wheel:** Rely on established solutions for handling financial calculations, which are more likely to have resolved common and obscure issues through community vetting.

Sample Fix:

To address precision and rounding issues, consider using a well-tested library like `bignumber.js` or `decimal.js` that is designed to handle arbitrary-precision decimal numbers and avoids common pitfalls of floating-point arithmetic:

```
import BigNumber from 'bignumber.js';

function format(big: string, denomination: number, decimals: number)
{
const number = new BigNumber(big);
const shifted = number.shiftedBy(-denomination);
return shifted.toFixed(decimals);
}
```

Resolution: This fixed function checks if the input string is an integer and, if `positiveNumbersOnly` is true, ensures it is non-negative.

```
import BigNumber from 'bignumber.js';

export const stringIsInteger = (
```

```
integer: string,
positiveNumbersOnly = true
) => {
const stringInteger = String(integer);
if (!stringInteger.match(/^[-]?\d+$/)) {
return false;
}
const bNparsed = new BigNumber(stringInteger);
const limit = positiveNumbersOnly ? 0 : -1;
return (
bNparsed.toString(10) === stringInteger && bNparsed.comparedTo(0) >=
limit
);
};
```

Evidences

Proof of Concept (PoC):

Location: src/denominate.ts

Reproduce:

In the `format` function, the handling of numbers as strings and manual insertion of decimal points without adequate consideration of floating-point arithmetic can lead to inaccuracies:

```
function format(big: string, denomination: number, decimals: number)
{
let array = big.toString().split('');
// Additional code for negative check and string manipulation
array.splice(array.length - denomination, 0, '.');
// Trimming and formatting logic
}
```

F-2024-2835 - Lack of Secure Transmission in API Calls - Info

Description: The function `getNetworkConfig` uses the `fetch` API to retrieve network configuration from a server without explicitly ensuring the use of secure protocols (HTTPS). This could potentially allow sensitive data to be transmitted over insecure channels, making it susceptible to interception or manipulation.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Recommendations

Remediation:

- **Enforce HTTPS:** Always use HTTPS for all external network calls to ensure encrypted transmissions.
- **URL Validation:** Implement rigorous validation of URLs to ensure they are secure and conform to expected formats and protocols.

Sample Fix:

Ensure that all API calls are made over HTTPS and consider implementing checks to verify the security of the protocol used in the URL.

```
export const getNetworkConfig = async (apiUrl: string): Promise<Netw
orkConfig | undefined> => {
if (!apiUrl.startsWith('https://')) {
throw new Error('Insecure connection protocol');
}
const response = await fetch(`${apiUrl}/network/config`);
...
};
```

Resolution:

```
export const getNetworkConfig = async (
apiUrl: string,
): Promise<NetworkConfig | undefined> => {
if (!apiUrl.startsWith('https://')) {
throw new Error('Insecure connection protocol');
}

try {
const response = await fetch(`${apiUrl}/network/config`);
```

Evidences

Proof of Concept (PoC):

Location: src/network.ts

Reproduce:

If the API calls are intercepted due to the use of an insecure connection (HTTP), sensitive data could be exposed or tampered with, which might lead to broader security implications for the network operations.

```
export const getNetworkConfig = async (apiUrl: string): Promise<Netw
orkConfig | undefined> => {
const response = await fetch(`${apiUrl}/network/config`);
...
};
```

F-2024-2836 - Insufficient Error Handling and Potential Data Leakage - Info

Description: The function `getNetworkConfig` catches errors broadly and returns `undefined` on any failure. This generic error handling could obscure the underlying cause of issues, potentially making debugging difficult and could inadvertently leak information through error messages.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Recommendations

Remediation:

- **Structured Error Handling:** Use structured error handling to provide more detailed internal diagnostics and avoid returning or logging any information that might help an attacker.
- **Logging and Monitoring:** Implement comprehensive logging for errors and monitor these logs to detect and respond to issues promptly.

Sample Fix:

Improve error handling to differentiate between types of errors and handle each appropriately. Log errors internally but avoid sending detailed error messages that might expose sensitive data.

```
catch (error) {
console.error('Failed to fetch network config:', error);
throw new Error('Network configuration retrieval failed');
}
```

Resolution:

```
catch (error) {
throw new Error('Failed to fetch network config');
}
```

Evidences

Proof of Concept (PoC):

Location: src/network.ts

Reproduce:

Poor error handling can lead to difficulties in maintaining security over time due to the inability to properly diagnose issues.

```
catch (error) {
return undefined;
}
```

F-2024-2991 - Caret Range Versioning Vulnerability in Dependency Management - Info

Description: The use of the caret (^) in versioning within the `package.json` file can introduce a risk of automatically upgrading to newer minor versions of dependencies that might include non-backward compatible changes or newly introduced vulnerabilities without thorough testing. This can lead to unexpected application behavior or security risks.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Recommendations

Remediation: Pin dependency versions to avoid unexpected updates that could introduce vulnerabilities.

Sample Fix:

To mitigate this issue, you can pin the dependency versions by removing the caret (^). This ensures that only the specified versions are used, thus avoiding unexpected updates. Modify the `package.json` as follows:

```
"dependencies": {
"@multiversx/sdk-core": "12.18.0",
"@multiversx/sdk-network-providers": "2.2.1",
"@multiversx/sdk-wallet": "4.3.0",
"buffer": "6.0.3"
}
```

Resolution:

```
"dependencies": {
"@multiversx/sdk-core": "13.1.0",
"@multiversx/sdk-network-providers": "2.2.1",
"@multiversx/sdk-wallet": "4.3.0",
"bignumber.js": "^9.1.2",
"buffer": "6.0.3",
"protobufjs": "7.3.0"
}
```

Evidences

Proof of Concept (PoC):

Location: package.json

Reproduce:

Consider the dependency `"@multiversx/sdk-core": "^12.18.0"` in the provided `**package.json**`. This configuration allows npm to install any version of `@multiversx/sdk-core` that is greater than or equal to `12.18.0` but less than `13.0.0`. If a new version `12.18.1` is released and contains a critical vulnerability or non-backward compatible changes, it will automatically be included in builds due to the caret range. This could compromise the security or functionality of the application without the developers' direct knowledge.

F-2024-2995 - Missing Author Information in Package Metadata - Info

Description: The absence of an author field in the `package.json` file of a Node.js package can lead to issues related to credibility, accountability, and traceability. While not a security vulnerability in the traditional sense, missing author information can decrease the transparency and trustworthiness of a package, particularly in public repositories where users evaluate the reliability and support of a package based on its metadata.

Assets:

- Metamask Snap [https://github.com/hknio/mx-metamask-snaps-d05bcac3ec10375973da6/]

Status: Fixed

Recommendations

Remediation: It is recommended to always include at least minimal contact information in the package metadata to enhance the credibility and trustworthiness of the package. Regular audits of package metadata should be conducted as part of the software development lifecycle to ensure that all necessary fields are accurately populated and up-to-date

Sample Fix:

To address this, populate the `author` field in the `package.json` file with relevant details. The author field can include a name, email, and url. Here is an example of how to structure it:

```
"author": {
"name": "Ben Dan",
"email": "Ben@hacken.io", //Optional
"url": "https://www.hacken.io"
}
```

Resolution:

```
"author": {
"name": "MultiversX",
"email": "extensions@multiversx.com",
"url": "https://multiversx.com"
},
```

Evidences

Proof of Concept (PoC):

Location: package.json

Reproduce: In the provided `package.json` for the package **`@multiversx/metamask-snap`**, the author field is observed to be empty:

```
"author": ""
```

Disclaimers

Hacken Disclaimer

The application given for audit has been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in the application's source code, its deployment, and functionality (performing the intended functions) being the focus of our analysis.

The report contains no statements or warranties regarding the identification of all vulnerabilities or the absolute security of the code. The report covers only the code that was submitted and reviewed, and therefore may not remain relevant after any modifications have been made. This report should not be considered a definitive or exhaustive assessment of the utility, safety, or bug-free status of the application, nor should it be taken as a guarantee of the absence of other potential issues.

While we have exerted our best efforts in conducting the analysis and producing this report, it is crucial to understand that this report should not be the sole source of reliance for ensuring the security of the application. We strongly recommend undertaking multiple independent audits and establishing a public bug bounty program to enhance the security posture of the application.

English is the original language of this report. The Consultant is not liable for any errors or omissions in any translations of this report.

Technical Disclaimer

Applications, whether decentralized apps (DApps) or other types of applications, are deployed and run within specific environments that may include various platforms, programming languages, and other related software components. These environments and components can have inherent vulnerabilities that might lead to security breaches. Consequently, the Consultant cannot guarantee the absolute security of the audited application.

Appendix 1. Severity Definitions

Severity	Description
Critical	These issues present a major security vulnerability that poses a severe risk to the system. They require immediate attention and must be resolved to prevent a potential security breach or other significant harm.
High	These issues present a significant risk to the system, but may not require immediate attention. They should be addressed in a timely manner to reduce the risk of the potential security breach.
Medium	These issues present a moderate risk to the system and cannot have a great impact on its function. They should be addressed in a reasonable time frame, but may not require immediate attention.
Low	These issues present no risk to the system and typically relate to the code quality problems or general recommendations. They do not require immediate attention and should be viewed as a minor recommendation.

Appendix 2. Scope

The scope of the project includes the provided repository:

Scope Details	
Repository	https://github.com/multiversx/mx-metamask-snaps
Commit	992c22e
Npm Package	https://www.npmjs.com/package/@multiversx/metamask-snap
Requirements	
Technical Requirements	