verichains

SECURITY AUDIT OF

STAVAX ACCOUNT

Private Report

Oct 29, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Pentesting	Penetration Testing is a security exercise where a cyber-security expert attempts to find and exploit vulnerabilities in a computer system.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Oct 29, 2024. We would like to thank the Stavax for trusting Verichains Lab in testing website. Delivering high-quality audits is always our top priority.

This penetration testing (pentest) focused on identifying security flaws in code and the design of the Stavax Account. The scope of the testing is limited to the websites provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit and penetration testing process, the audit team had identified some issues in the source code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Stavax Account

Stavax Account is a self-custodial wallet that delivers seamless Web3 functionality through a user-friendly interface, featuring robust security and multi-chain integration.

Stavax Account includes all essential crypto wallet features, such as wallet connect, deposit, withdrawal, and token transfers, along with integration with various dApps.

In addition to a traditional wallet, Stavax Account offers an advanced Smart Wallet powered by MPC technology. While not supported by all networks, the Smart Wallet provides enhanced management, backup, restoration, and greater flexibility for transaction fees.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of the Stavax Account frontend and backend services.

It was conducted on commits:

- `29f4306ca7849e8bfb036634ceb44d589da2d7c9` on *https://git.xantus.network/stavax/stavax-frontend*
- `ac74d02921ba503b0b3b322f09473c38cf0de1cf` on *https://github.com/verichains/smartcontract-audit-v2/tree/master/projects/starvax*

The staging environment was used for the audit. The staging environment is a test environment that is a replica of the production environment. The staging frontend is located at *https://t.me/stv_staging_bot/app* and the backend service is deployed at *https://api.stavax-account.dev.xantus.dev*.

1.3. Audit Methodology

Our penetration testing (pentest) process for website includes two steps:

- Website are scanned/tested for commonly known and more specific vulnerabilities using security pentest tool.
- Manual pentest for security issues. The website is manually tested to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during testing:

For Client side:

- Broken Client-side Access Control
- DOM-based XSS

- Sensitive Data Leakage
- Lack of Third-party Origin Control
- JavaScript Drift
- Sensitive Data Stored Client-Side
- Client-side Security Logging and Monitoring Failures
- Not Using Standard Browser Security Controls
- Including Proprietary Information on the Client-Side

For Server side:

- Broken Access Control
- Cryptographic Failures
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

For API:

- Broken Object Level Authorization
- Broken Authentication
- Broken Object Property Level Authorization
- Unrestricted Resource Consumption
- Broken Function Level Authorization
- Unrestricted Access to Sensitive Business Flows
- Server Side Request Forgery
- Security Misconfiguration
- Improper Inventory Management
- Unsafe Consumption of APIs

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Stavax acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Stavax understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Stavax agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Stavax will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Stavax, the final report will be considered fully accepted by the Stavax without the signature.

This report contains sensitive information or information that's not meant to be for the general public. These will be censored out as requested by the Stavax and will be displayed as [redacted].

2. AUDIT RESULT

2.1. Overview

Stavax Account contains two main parts: the frontend client and the backend services.

2.1.1. The frontend client

The frontend client is primarily developed in `TypeScript` using the `Vue` framework. It integrates with Telegram and interacts with the backend service via API.

2.1.2. The backend service

The backend service is built in `Go`, utilizing the `Gin` framework to handle HTTP requests. It also employs RPC calls to retrieve network status and execute transactions.

2.2. Findings

During the audit process, the audit team had identified some issues in the source code.

2.2.1. INFORMATIVE - Wrong return data in `/transactions/history` endpoint

The `/transactions/history` returns the history of all transactions performed by users. However, the developer forgot to set the property `total_items` when creating the response. This ambiguous response may confuse the developers and causes errors in the future.

```
func (s *ServiceImpl) History(ctx context.Context, req HistoryReq) (*HistoryRes, error) {
    filter := repo.HistoryFilter{Address: req.Address}
    histories, total, err := s.transactionRepo.History(ctx, req.AuthUserID, filter,
req.PagingParam)
    if err != nil {
        return nil, err
    }

    return &HistoryRes{
        PagingResponse: req.PagingParam.ToResponse(total),
        Items:          lo.Map(histories, toItemRes),
    }, nil
}
```

RECOMMENDATION

Correct the value of `total_items` property in response message.

2.2.2. INFORMATIVE - Multiple unused API endpoint

During the audit process, we notice that the following APIs are not used and are not mention in the frontend's codebase:

- `/contacts`
- `/contacts/{id}`
- `/paymaster/particle/after-sign`
- `/paymaster/particle/before-sign`
- `/remote-configs/{app}/{key}`
- `/user-wallets/edit-name`
- `/user-wallets/import`

RECOMMENDATION

Double check these APIs and remove unnecessary endpoints.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Oct 29, 2024*	Private Report	Verichains Lab

Table 2. Report versions history