ShareX
Audit Report

Mon Jul 21 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

ShareX Audit Report

1 Executive Summary

1.1 Project Information

Description	ShareX is building the Web3 Consumer and Financial Layer for the sharing economy. By integrating IoT sharing service terminals with the Deshare Protocol, RWA solutions, and crypto payments, ShareX connects Web3 with global sharing economy brands, driving deep engagement and conversion of massive Web2 users and consumer scenarios
Type	Synthetic Assets
Auditors	ScaleBit
Timeline	Fri Jul 11 2025 - Mon Jul 21 2025
Languages	Solidity
Platform	BSC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/sharex-org/sharex-evm-contracts/
Commits	c3d88e161fd58d207133fc60c8acbd32dc3fb4b9 66f47d54a9c25fb688ff5569300d73c5272313c0

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
EVE	src/libraries/Events.sol	20a5c50a38c86a0b970a3f35ceba9a938d08e39e
CON	src/libraries/Constants.sol	1020e34a07a1f7ec17a4d22cf562c1a4096975d5
DTY	src/libraries/DataTypes.sol	f7d13570f5c8b6fc8a6cf6bae71ece0d87c8eed4
ERR	src/libraries/Errors.sol	0484d7957a4597e73300a1d6a969bcab063a9f77
DVA	src/DeshareVault.sol	9b92dff80660ab92afed68814ab69f91cdca29f2

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	2	2	0
Informational	1	1	0
Minor	1	1	0
Medium	0	0	0
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by ShareX to identify any potential issues and vulnerabilities in the source code of the ShareX smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 2 issues of varying severity, listed below.

ID	Title	Severity	Status
DVA-1	Redundant State Initialization in Constructor	Minor	Fixed
DVA-2	Missing Length Validation for Partner Name and Description	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the ShareX Smart Contract :

Admin

- `initialize` : Proxy contract initialization

OPERATOR

- `registerPartner` : Register new partners and update partner statistics
- `registerMerchant` : Register a new merchant
- `registerDevice` : Register a new device
- `uploadTransactionBatch` : Upload the compressed transaction batch data
- `registerCountry` : Register in a new country

4 Findings

DVA-1 Redundant State Initialization in Constructor

Severity: Minor

Status: Fixed

Code Location:

src/DeshareVault.sol#95-118

Descriptions:

The contract's `constructor` initializes several state variables, including `_roles`, `_version`, and various `_counters`. This same initialization logic is duplicated in the `initialize` function.
In an upgradeable contract pattern, the `constructor` is only executed once when the implementation contract is deployed. Its state is stored within the implementation contract itself. The `initialize` function, however, is called via `delegatecall` from the proxy, and it sets the state in the proxy's storage context.
Since all user interactions occur through the proxy, the state variables set in the `constructor` are never used by the live system. This leads to several issues:

1. **Gas Inefficiency:** The state-setting operations (`_grantRole`, `_version` assignment, `_counters` assignments) in the `constructor` consume deployment gas without providing any functional benefit to the proxied contract.
2. **Misleading Events:** The `ContractInitialized` event emitted from the constructor is misleading, as the actual system initialization happens when the `initialize` function is called on the proxy.

While the contract correctly calls `_disableInitializers()` to prevent malicious initialization of the implementation contract, the redundant setup code should be removed to adhere to best practices.

Suggestion:

It is recommended to remove the redundant state-setting logic from the constructor.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

DVA-2 Missing Length Validation for Partner Name and Description

Severity: Informational

Status: Fixed

Code Location:

src/DeshareVault.sol#174-175

Descriptions:

In the `registerPartner()` function, the protocol sets partner information such as `partnerId`, `partnerCode`, `partnerName`, and `description`.

```
PartnerInfo storage partner = _partners[partnerId];
        partner.id = partnerId;
        partner.partnerCode = params.partnerCode;
        partner.verification = params.verification;
        partner.timestamp = uint32(block.timestamp);
        partner.iso2 = params.iso2;
        partner.partnerName = params.partnerName;
        partner.description = params.description;
        partner.businessType = params.businessType;

        _partnerCodeToId[partnerCodeHash] = partnerId;
```

However, the protocol does not validate the length of `partnerName` and `description`. If these fields are excessively long, it may lead to unexpected behavior or issues in storage, event logs, or front-end display.

Suggestion:

It is recommended to add proper length validation for `partnerName` and `description` to ensure they remain within safe and expected limits.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.