SALUS SECURITY

NOV 2024

CODE SECURITY ASSESSMENT

SOLV FINANCE

Overview

Project Summary

- Name: Solv Finance - solvBTC
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/solv-finance/SolvBTC
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Solv Finance - solvBTC
Version	v2
Type	Solidity
Dates	Nov 28 2024
Logs	Nov 25 2024; Nov 28 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	1
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Emit unexpected event	Low	Logging	Resolved
2	Redundant Code	Informational	Redundancy	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Emit unexpected event

Severity: Low	Category: Logging
Target: - contracts/SolvBTCRouterV2.sol	

Description

When calling `removeKycSBTVerifier()`, the event is still triggered even if the parameter `kycSBTVerifier_` does not exist in the `kycSBTVerifiers` array.

contracts/SolvBTCRouterV2.sol:L218 - L227

```
    function removeKycSBTVerifier(address kycSBTVerifier_) external onlyOwner {
        for (uint256 i = 0; i < kycSBTVerifiers.length; i++) {
            if (kycSBTVerifiers[i] == kycSBTVerifier_) {
                kycSBTVerifiers[i] = kycSBTVerifiers[kycSBTVerifiers.length - 1];
                kycSBTVerifiers.pop();
                break;
            }
        }
        emit RemoveKycSBTVerifier(kycSBTVerifier_);
    }
```

Recommendation

Consider moving the event emission into the `if` statement.

```
    function removeKycSBTVerifier(address kycSBTVerifier_) external onlyOwner {
        for (uint256 i = 0; i < kycSBTVerifiers.length; i++) {
            if (kycSBTVerifiers[i] == kycSBTVerifier_) {
                kycSBTVerifiers[i] = kycSBTVerifiers[kycSBTVerifiers.length - 1];
                kycSBTVerifiers.pop();
                emit RemoveKycSBTVerifier(kycSBTVerifier_);
                break;
            }
        }
    }
```

Status

The team has resolved this issue in commit 995148c.

2.3 Informational Findings

2. Redundant Code

Severity: Informational	Category: Redundancy
Target: - contracts/SolvBTCRouterV2.sol	

Description

Unused code should be removed before deploying the contract to mainnet. We have identified the following code are not being utilized:

1. The `console` contract is useful for testing purposes but has no practical use in the actual project application.

contracts/SolvBTCRouterV2.sol:L15

```
import "../lib/forge-std/src/console.sol";
```

2. The `checkKycSBT()` function is called in the `deposit()` function to check the identity of the caller, and the `checkPoolPermission()` function is called in the `_deposit()` function, but as long as the `checkKycSBT()` function returns true, the return value of the `checkPoolPermission()` function is always `true`.

Recommendation

Consider removing the redundant code.

Status

The team has resolved this issue in commit 995148c.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit c8a09fa :

File	SHA-1 hash
contracts/SolvBTCRouterV2.sol	ce687b0428efd4b742894b66d1dde129b6da4db0