SALUS SECURITY

FEB 2023

CODE SECURITY ASSESSMENT

xDonations

Overview

Project Summary

- Name: xDonations
- Version: commit 6710b98
- Platform: evm-compatible chains
- Language: Solidity
- Repository: https://github.com/connext/xDonations
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	xDonations
Version	v1
Type	Solidity
Dates	Feb 10 2023
Logs	Feb 10 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	2
Total informational issues	1
Total	4

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

Salus Security is an all-rounded blockchain security company. With rich experiences in traditional and blockchain security, we are born to solve some of the most complex security issues in the industry and make security services accessible for all. Our smart contract auditing service is equipped with an automated tool and expert services. Every project needs an invincible shield to achieve long-term success; with complete coverage from traditional to blockchain, Salus Security is what you need. 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	Malicious sweeper can DOS contract by removing all sweepers	Medium	Business Logic	Unresolved
2	Mismatch between comment and implementation	Low	Configuration	Unresolved
3	Use of a function that is outside the audit scope	Low	Audit scope	Unresolved
4	Use of floating compiler version	Informational	Configuration	Unresolved

2.2 Notable Findings

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

1. Malicious sweeper can DOS contract by removing all sweepers	
Severity: Medium	Category: Business Logic
Target: - contracts/xDonate.sol	

Description

In the xDonate contract, there is a privileged **sweeper** role.

The sweeper of the xDonate contract:
- can add or remove sweepers
- can call sweep() to move funds from this contract to the donationAddress

It's worth noting that
- only the sweeper can move funds out from this contract using sweep()
- only sweeper can add or remove sweepers

So a malicious sweeper can remove all the sweepers, resulting in a situation where sweep() can not be called successfully and all funds are locked in the contract.

Recommendation

Consider adding an owner role that manages sweepers.

2. Mismatch between comment and implementation

Severity: Low	Category: Configuration
Target: - contracts/xDonate.sol	

Description

contracts/xDonate.sol:L43

```
uint256 public constant MIN_SLIPPAGE = 10; // 0.01% is min slippage
```

The comment indicates that the minimum slippage is 0.01%, however, the implementation (`MIN_SLIPPAGE = 10`) means the minimum slippage is 0.1%.

Recommendation

Consider fixing the mismatch.

3. Use of a function that is outside the audit scope

Severity: Low	Category: Audit scope
Target: - contracts/xDonate.sol	

Description

contracts/xDonate.sol:L202-L210

```
bytes32 transferId = connext.xcall{value: msg.value}(
    donationDomain,         // _destination: Domain ID of the
destination chain
    donationAddress,        // _to: address receiving the funds on the
destination
    donationAsset,          // _asset: address of the token contract
    msg.sender,             // _delegate: address that can revert or
forceLocal on destination
    amountOut,              // _amount: amount of tokens to transfer
    connextSlippage,        // _slippage: the maximum amount of slippage
the user will accept in BPS
    bytes("")               // _callData: empty bytes because we're only
sending funds
);
```

The xDonate contract uses the xcall() function from the connext protocol to transfer funds across chains. However, the code in xcall() is not within the scope of the current audit. The xcall() function was assumed functioning and secure in the current audit.

2.3 Informational Findings

4. Use of floating compiler version

Severity: Informational	Category: Configuration
Target: - contracts/xDonate.sol - contracts/interfaces/IWeth.sol	

Description

contracts/xDonate.sol:L2

```
pragma solidity ^0.8.17;
```

The xDonate contract uses a floating compiler version ^0.8.17.

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

Consider locking the pragma version to 0.8.17.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 6710b98:

File	SHA-1 hash
contracts/xDonate.sol	fe451c1bf89c688412ebc598ac5c09e8e169a3d1
contracts/interfaces/IWeth.sol	0a997998f4048211b9b951d89205c8762a749434