macro

xDonations A-1

Security Audit

February 10, 2023
Version 1.0.0

Presented by 0xMacro

Table of Contents

Introduction

This document includes the results of the security audit for xDonations's smart contract code as found in the section titled 'Source Code'. The security audit was performed by the Macro security team on February 10, 2023.

The purpose of this audit is to review the source code of certain xDonations Solidity contracts, and provide feedback on the design, architecture, and quality of the source code with an emphasis on validating the correctness and security of the software in its entirety.

Disclaimer: While Macro's review is comprehensive and has surfaced some changes that should be made to the source code, this audit should not solely be relied upon for security, as no single audit is guaranteed to catch all possible bugs.

Overall Assessment

The following is an aggregation of issues found by the Macro Audit team:

Severity	Count	Acknowledged	Won't Do	Addressed
High	2	-	-	2
Low	1	1	-	-
Code Quality	1	-	-	1
Informational	1	-	-	-
Gas Optimization	1	-	-	1

xDonations was quick to respond to these issues.

Specification

Our understanding of the specification was based on the following sources:

- Discussions on Telegram with the xDonations team.

Source Code

The following source code was reviewed during the audit:

- **Repository:** xDonations
- **Commit Hash:** `ca30a0937c37c9e30609cae6da6464b0559cf465`

Specifically, we audited the following contracts within this repository:

Contract	SHA256
contracts/xDonate.sol	`544c2e083816a09ccd0a951eba8e4c5a846fad55787561e72b5dfcd744f46884`

Note: This document contains an audit solely of the Solidity contracts listed above. Specifically, the audit pertains only to the contracts themselves, and does not pertain to any other programs or scripts, including deployment scripts.

Issue Descriptions and Recommendations

Click on an issue to jump to it, or scroll down to see them all.

- ~~H-1~~ Slippage calculation is incorrect
- ~~H-2~~ Default slippage is set to a high number
- L-1 The bridge has unlimited approvals for `donationAsset`
- ~~Q-1~~ Incorrect comment giving the wrong impression to the reader
- ~~G-1~~ Token approval could be moved to the constructor to eliminate redundant `SLOADs`
- I-1 `_sweep()` does not check for max slippage

Security Level Reference

We quantify issues in three parts:

1. The high/medium/low/spec-breaking **impact** of the issue:
    - How bad things can get (for a vulnerability)
    - The significance of an improvement (for a code quality issue)
    - The amount of gas saved (for a gas optimization)
2. The high/medium/low **likelihood** of the issue:
    - How likely is the issue to occur (for a vulnerability)
3. The overall critical/high/medium/low **severity** of the issue.

This third part – the severity level – is a summary of how much consideration the client should give to fixing the issue. We assign severity according to the table of guidelines below:

Severity	Description
(C-x) Critical	We recommend the client **must** fix the issue, no matter what, because not fixing would mean **significant funds/assets WILL be lost.**
(H-x) High	We recommend the client **must** address the issue, no matter what, because not fixing would be very bad, *or* some funds/assets will be lost, *or* the code's behavior is against the provided spec.
(M-x) Medium	We recommend the client to **seriously consider** fixing the issue, as the implications of not fixing the issue are severe enough to impact the project significantly, albiet not in an existential manner.
(L-x) Low	The risk is small, unlikely, or may not relevant to the project in a meaningful way. Whether or not the project wants to develop a fix is up to the goals and needs of the project.
(Q-x) Code Quality	The issue identified does not pose any obvious risk, but fixing could improve overall code quality, on-chain composability, developer ergonomics, or even certain aspects of protocol design.
(I-x) Informational	Warnings and things to keep in mind when operating the protocol. No immediate action required.
(G-x) Gas Optimizations	The presented optimization suggestion would save an amount of gas significant enough, in our opinion, to be worth the development cost of implementing it.

Issue Details

~~H-1~~ Slippage calculation is incorrect

TOPIC	STATUS	IMPACT	LIKELIHOOD
Sandwich Attack	Fixed	High	High

`amountOutMInium` is calculated incorrectly for uniswap swap inside `_sweep`.

```
uint256 amountInNormalized = normalizeDecimals(IERC20Metadata(fromAsset).c

// Set up uniswap swap params.
ISwapRouter.ExactInputSingleParams memory params =
  ISwapRouter.ExactInputSingleParams({
      tokenIn: fromAsset,
      tokenOut: donationAsset,
      fee: poolFee,
      recipient: address(this),
      deadline: block.timestamp,
      amountIn: amountIn,
      amountOutMinimum: amountInNormalized * (10_000 - uniswapSlippage) /
      sqrtPriceLimitX96: 0
  });
```

It's being derived directly from `amountIn` using decimals and slippage, instead of a current market price from uniswap pool.

Hence it would either overvalue the amount expected or undervalue it.

https://docs.uniswap.org/contracts/v3/guides/swaps/single-swaps

In the case of overvalued, it would revert only, while in the case of undervalued, it would allow a sandwich attack.

Remediations to consider

Consider calculating `amountOutMinium` off-chain using any price oracle or `uniswap` quoter and making `amountOutMinium` a user input.

~~H-2~~ Default slippage is set to a high number

TOPIC	STATUS	IMPACT	LIKELIHOOD
Sandwich Attack	Fixed	High	High

```
function sweep (
        address fromAsset,
        uint24 poolFee,
        uint256 amountIn
    ) external payable onlySweeper {
        _sweep (
            fromAsset,
            poolFee,
            amountIn,
            1000, // 1% default max slippage
            100 // 1% default max slippage
        );
    }
```

`sweep()` sets slippage for swap to `1_000`, and the comment states that slippage is intended to be set to 1%.

```
amountOutMinimum: amountInNormalized * (10_000 - uniswapSlippage) / 10_000
```

However, the calculation is performed using `10_000` as a precision basis, and this causes slippage to be calculated as 10%, which is a huge slippage and will make swaps vulnerable to sandwich attacks.

Remediations to consider

Use `100` as the default slippage.

L-1 The bridge has unlimited approvals for `donationAsset`

TOPIC	STATUS	IMPACT	LIKELIHOOD
Trust Model	Acknowledged	High	Low

The bridge is given unlimited token transfer approvals. Although the bridge is a trusted contract, it is recommended to give only required approvals to external parties since any potential bugs or vulnerabilities in the bridge may cause a loss of assets in the donation contract.

```
if (!approvedDonationAsset) {
    approvedDonationAsset = true;
    // use max approval for assset
    TransferHelper.safeApprove(donationAsset, address(connext), MAX_INT);
}
```

Remediations to consider

Remove `MAX_INT` approval and approve only the required amounts.

Q-1 Incorrect comment giving the wrong impression to the reader

TOPIC	STATUS	QUALITY IMPACT
Comments	Fixed	Low

```
address public immutable donationAsset; // should be USDC
```

The readers of the contracts can assume that donations could be made through USDC only, consider removing this comment.

G-1 Token approval could be moved to the constructor to eliminate redundant `SLOADs`

TOPIC	STATUS	GAS SAVINGS
Gas	Fixed	Low

```
// Approve connext to bridge donationAsset.
if (!approvedDonationAsset) {
    approvedDonationAsset = true;
    // use max approval for assset
    TransferHelper.safeApprove(donationAsset, address(connext), MAX_INT);
}
```

Every `_sweep()` reads the storage variable `approvedDonationAsset` , to verify if there is a need to do a token approval. Instead, approval for `donationAsset` could be moved to the constructor to avoid these redundant reads.

I-1 `_sweep()` does not check for max slippage

TOPIC	IMPACT
Sandwich Attack	Informational ✳

`sweep()` is a function that can be called with any slippage, and `_sweep()` only checks for the `MIN_SLIPPAGE`. Since this function expects parameters with a precision basis, it is prone to human errors(see H-2) and a wrong slippage passed here may lose funds to sandwich attacks.

Adding an upper limit for slippages, similar to `MIN_SLIPPAGE`, would resolve this issue; but that may block the sweeper from swapping tokens with low liquidity and is not recommended if such tokens are expected to be handled. Calling `sweep()` through a client app is recommended to minimize human errors.

Disclaimer

Macro makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and Macro specifically disclaims all implied warranties of merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

Macro will not be liable for any lost profits, business, contracts, revenue, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand by any other party. In no event will Macro be liable for consequential, incidental, special, indirect, or exemplary damages arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if Macro has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the Emergent team and only the source code Macro notes as being within the scope of Macro's review within this report. This report does not include an audit of the deployment scripts used to deploy the Solidity contracts in the repository corresponding to this audit. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. In this report you may through hypertext or other computer links, gain access to websites operated by persons other than Macro. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such websites' owners. You agree that Macro is not responsible for the content or operation of such websites, and that Macro shall have no liability to your or any other person or entity for the use of third party websites. Macro assumes no responsibility for the use of third party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.