LIDO-DAO WSTETH SMART CONTRACT AUDIT

September 07, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Lido-dao. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

LIDO protocol is a project for stacking Ether to use it in Beacon chain. Users can deposit Ether to the Lido smart contract and receive stETH tokens in return. The stETH token balance corresponds to the amount of Beacon chain Ether that the holder could withdraw if state transitions were enabled right now in the Ethereum 2.0 network.

1.3 SECURITY ASSESSMENT METHODOLOGY

A group of auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code
> Checking with static analyzers (i.e Slither, Mythril, etc.)

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the code for compliance with the desired security model:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> Cross-check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level and provide the client with an interim report.

05 Bug fixing & re-check:
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

LIDO WstETH smart contract gives opportunity for users to use their stETH assets in various defi protocols, because wstETH balance for each user in stable and not changing through time unlike stETH balance.

1.5 PROJECT DASHBOARD

Client	Lido-dao
Audit name	WstETH
Initial version	ea6fa222004b88e6a24b566a51e5b56b0079272d
Final version	-
Date	August 18, 2021 - September 07, 2021
Auditors engaged	2 auditors

FILES LISTING

WstETH.sol	https://github.com/lidofinance/lido-dao/tree/ea6fa222004b88e6a24b566a51e5b56b0079272d/contracts/0.6.12/WstETH.sol
IStETH.sol	https://github.com/lidofinance/lido-dao/tree/ea6fa222004b88e6a24b566a51e5b56b0079272d/contracts/0.6.12/interfaces/IStETH.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	5
Comment	0

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical or major issues were found. Several findings were marked as warnings, after working on the reported issues all of them were acknowledged or agreed as no issues (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `-`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Possible incorrect initialization
File	WstETH.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

In the following function `_stETH` can be zero address:
WstETH.sol#L39

RECOMMENDATION

We recommend to add the following check:

```
address(_stETH) != address(0), "wstETH: incorrect address");
```

CLIENT'S COMMENTARY

We acknowledge the issue. The current WstETH instance (0x7f39c581f595b53c5cb19bd0b3f8da6c935e2ca0) has the correct StETH contract address (0xae7ab96520de3a18e5e111b5eaab095312d7fe84).

WRN-2	Working with values equal to zero
File	WstETH.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

`wstETHAmount` or `stETHAmount` potentially can be zero:
WstETH.sol#L56
WstETH.sol#L73

RECOMMENDATION

We recommend to add the following check:

```
require(wstETHAmount > 0, "wstETH: stETH return 0 shares");
```

CLIENT'S COMMENTARY

We acknowledge the issue. The case is possible for wrap calls with small values (e.g. several Weis) then `stEthPerToken` is greater than 1. Currently `stEthPerToken` is 1.04, and wrapping 1 stETH Wei will return 0 wstETH Weis. However, these losses are negligible compared to the gas cost for the transaction, and don't lead to violation of the token's mechanics. We would add this edge case to the documentation.

WRN-3	stETH can be paused
File	WstETH.sol
Severity	Warning
Status	**No Issue**

DESCRIPTION

`stETH` can be paused, so all transfers would revert:
WstETH.sol#L57
WstETH.sol#L73
WstETH.sol#L81

RECOMMENDATION

We recommend to add the following check (for this check it is necessary to update interface):

```
require(!stETH.isStopped(), "wstETH: transfer stopped");
```

CLIENT'S COMMENTARY

The proposed improvement will slightly increase the readability of the revert message by adding a small gas and contract size overhead. We believe that the current revert message (CONTRACT_IS_STOPPED) is an acceptable compromise in this situation.

WRN-4	Unchecked returned value
File	WstETH.sol
Severity	Warning
Status	**No Issue**

DESCRIPTION

Returned value for `transfer`, `transferFrom` not checked:
WstETH.sol#L57
WstETH.sol#L73

RECOMMENDATION

We recommend to add a check like this:

```
require(stETH.transferFrom(msg.sender, address(this), _stETHAmount), "wstETH: transfer fails
```

CLIENT'S COMMENTARY

Since the StETH's `transfer` and `transferFrom` methods always revert instead of returning `false`, we are not checking them for gas-saving reasons. The improvement might slightly increase the readability of the error, but would add some overhead on contract size. Besides, StETH token already has good error descriptions like `TRANSFER_FROM_THE_ZERO_ADDRESS`, `TRANSFER_TO_THE_ZERO_ADDRESS`, and `TRANSFER_AMOUNT_EXCEEDS_BALANCE`.

WRN-5	Incorrect burning of shares
File	WstETH.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Burning of shares for `WstETH` contract from `stETH` contract can lead to block `unwrap` function for users:
WstETH.sol#L69-L75

RECOMMENDATION

We recommend to add a check to `Lido` contract, that Burner can't burn shares for `WstETH`.

CLIENT'S COMMENTARY

Any burning of the stETH token is an emergency that Lido DAO reserves to use against protocol hack or to recover from a failure mode. The burning of tokens for an arbitrary address shouldn't happen during normal protocol operations. We acknowledge, that burning any number of stETHs on the WstETH contract balance will violate the wrapping/unwrapping mechanics, but this shouldn't happen in a normal mode. However, this opportunity is very important for failure recovery: if there is an error in the current implementation of the WstETH token, then the DAO will be able to pause StETH, burn stETH from the WstETH contract's balance, redeploy a new token, and recover balances by minting after that.

3.ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

Ethereum
Cosmos
EOS
Substrate

TECH STACK

Python
Solidity
Rust
C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes