wSHIFT Token Security Audit

by Monte Labs

December 16, 2020

1 Introduction

This document is a security audit report for for the Wrapped Shift (wSHIFT) smart contracts, where we discuss design, architecture and potential issues found while analyzing and testing the code.

The contracts use a few OpenZeppelin libraries, which are considered standard and have been heavily audited and used by the community in general. The code is easy to read, well documented and tested exhaustively as 100% of the contract is covered by unit tests. We did not find any issues and our concerns are purely architectural. Those are detailed in Section 5. Section 6 presents our concluding thoughts on the audit.

2 Code Review Overview

2.1 Methodology of the Audit

Our methodology in performing the code review consisted of four chronologically executed phases:

1. Understanding the existing documentation, purpose and specifications of the smart contract.
2. Executing automated tools to scan for generic security vulnerabilities.
3. Manual analysis covering both functional (best effort based on the provided documentation) and security aspects of the smart contracts.
4. Writing this report.

2.2 Limitation

Security audits do not uncover all existing vulnerabilities; even an audit in which no vulnerabilities are found such as this one is no guarantee of secure smart contracts. Our audits enable discovery of vulnerabilities that were overlooked during development and have been proven effective finding vulnerabilities that posed multiple security risks. We have extensively reviewed the smart contracts in the scope of this audit using a variety of techniques mentioned above to ensure the expected execution of the audited smart contracts.

3 Scope of Code Review

The code to be audited can be found at github.com/ShiftNrg/wrappedShift.

Source code files received	September 21, 2020
Git commit	9a4a8f74e348a2015ab63f3ea1e6db2636104858
EVM version	istanbul
Initial Compiler	SOLC compiler, version 0.6.12

The scope of the audit is limited to the following source code files.

3.1 Impacted files

File	SHA-1 checksum
WrappedShift.sol	5f195d7fd2b3089a06b051df1263d8d7cb8a4462
ERC20Capped.sol	f9ce65035fdf845b34cb51e02cdb8eae5869d8ec

4 Token overview

Wrapped Shift (wSHIFT) is a ERC-20 Token which is capped, pausable and burnable. It derives most of these functionalities from OpenZeppelin's smart contracts with a modification that allows a specific user/contract to increase wSHIFT's token cap. The token defines roles to manage the contract and control functions related to burning tokens, defining and extending token caps, minting new tokens and pause the token.

5 Issues

We did not find issues regarding the SHIFT's token contracts. Below we outline some design decisions that can lead to centralization but adhere perfectly to the proposed platform architecture outlined in the main repository at github.com/ShiftNrg.

5.1 Token Design and Architecture

Pausable tokens. Tokens that can be pausable might centralize the system if the entity in power of pausing the smart contract is centralized or even an account. A governance smart contract can be used if the feature needs to be implemented.

Unlimited cap. This design decision might cause same concerns as the previous one, and it might become a single point of centralization. Again, using a governance protocol to decide these token attributes might be the best solution for continuing with the proposed design.

5.2 Minor Issues

The only minor issue we would like to highlight is a misleading comment in the `dev` NatSpec tag for the `canBurn` modified written in `WrappedShift.sol`. The comment says `Throws if called by any account other than the owner`, but that is not what the modifier does. It throws if no token admin has set the `burningEnabled` flag to `true`.

6 Conclusion

We have analyzed the wSHIFT smart contracts regarding security and architecture. The code provided is exceptionally written and easy to read, it has a good documentation and specification and 100% test coverage by unit tests. We did not find any issues related to the smart contract as they are specified in the documentation, and we offer some suggestions to avoid centralization.

This document is stored on IPFS as a security evidence. This audit also implies that MonteLabs issues an on-chain audit verification stamp that can

be accessed directly via the smart contract or at https://montelabs.com/audits as soon as the contracts are deployed, where all the security evidences can be fetched.