Smart contract security audit

HolderFinance [Bridge Protocol]

v.1.0

Table of Contents

1.0 Introduction

1.1 Project engagement

During April of 2021, HolderFinanceengaged CTDSec to audit smart contracts that they created, in the last version the base contracts were revised and in this version the farming part is specifically revised. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. HolderFinanceprovided CTDSec with access to their code repository and whitepaper.

HFi is a project initiated within the Ethereum network. It aims to become the best store of value tokens benefiting from the Ethereum network advanced technology, and to be the first Holders centric project of the Cross Chain DeFi ecosystem. It rewards early adopters and Holders of HFi who stake this community-driven token. Indeed, the longer you stake HFi, the more you are rewarded by earning more HFi. The very limited total supply combined with the huge potential of the cross chain ecosystem converge to develop the decentralized finance (DeFi) exposure. The purpose of HFi is to become a digital store of value, like a digital gold combining the fast transaction speed a metadata structure of its main product “HolderSwap'' to tackle the high gas cost fees on Ethereum thus, allows to anyone worldwide to be part of HFi project and build their future wealth. HFi is initiated and developed with the community in mind and for the purpose of rewarding its members proportional to their investment.

1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that HolderFinanceteam put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the HolderFinancecontract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

bridge_SC.sol - f27ba7c9b92bec0080441be0b14b4f9648dca91a7a53119d492e690c99f4347a [SHA256]

2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

№	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	PASSED
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED
15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED

17	Arithmetic accuracy.	PASSED
18	Design Logic.	PASSED
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	PASSED

3.0 Security Issues

3.1 High severity issues [1 - Solved]

1. Minting issue

Issue

There is no checking that the nonce in mint function will be less or equal to the nonce used in burn function, so the owner will be able to mint tokens to any address using any number greater than current nonce, so every future mints will fail. Also there is no checking that to addresses in burn and mint functions are the same for similar nonce value. In this case there could be minting to another address, instead of real to address provided in burn function.

Recommendation:

Please check the logic of this function, and check that nonce could not be greater in mint function than the last nonce value used in burn. Also check that to addresses are the same for the same nonce values.

Comment:

This was a decision of architecture that needed to have a single nonce because it needed to have multi requests/processes in different timeframes.

Issue is solved after reviewing with the development team.

3.2 Medium severity issues [1 - Solved]

1. Wrong from address in event

Issue:

In mint function there will be wrong from address in Transfer event, because from address always will be owner, not the real burner.

Recommendation:

It would be better to write there real from address.

Comment:

After reviewing with the development team we found that the minter is the bridge script who has administrative rights on the bridge and to is the destination address of the token who burn it on the other blockchain.

Issue is solved after reviewing with the development team.

3.3 Low severity issues [0]

No low security issues found.

4.0 Functions Outline

+ [Int] IToken

- [Ext] mint #

- [Ext] burn #

- [Ext] burnFrom #

+ Ownable

- [Pub] #

- [Ext] owner

- [Pub] isOwner

- [Ext] renounceOwnership #

- modifiers: onlyOwner

- [Ext] transferOwnership #

- modifiers: onlyOwner

- [Int] _transferOwnership #

+ BridgeBase (Ownable)

- [Pub] <constructor> #

- [Ext] burn #

- [Ext] mint #

- modifiers: onlyOwner

+ BridgeBsc (BridgeBase)

- [Pub] <constructor> #

- modifiers: BridgeBase

+ BridgeEth (BridgeBase)

- [Pub] <constructor> #

- modifiers: BridgeBase

($) = payable function

 # = non-constant function

5.0 Summary of the audit

Contract is safe to deploy.