Holdefi Audit

MARCH 29, 2021 | IN SECURITY AUDITS | BY OPENZEPPELIN

```
L
U
S
I
O
N
```

Update:
OpenZeppelin completed the audit of the Holdefi platform and delivered the findings in April 2020, but the publishing of the report has been delayed until March 2021. As the DeFi ecosystem has evolved over the last year, OpenZeppelin strongly recommends another audit be conducted

on the codebase prior to a mainnet launch.

The fixes to the issues identified in this report have not been reviewed by OpenZeppelin.

Introduction

Holdefi is a lending platform where users can hold their assets and earn interest or borrow tokens and repay them after a specific

period of time. Anyone can supply assets to Holdefi's liquidity pool and immediately begin earning interest.

The Holdefi team asked us to review and audit the smart contracts for their DeFi protocol. We examined the code, and here we publish our findings.

The audited commit is `f4df394d7cf6df347b1f1e9af8e2676c5933c2c9` and the

files included in the scope were Holdefi, HoldefiPauser, HoldefiPrices, HoldefiSettings, CollateralsWallet, and Ownable. The SafeMath, SampleToken and SimpleMedianizer were not included in the scope.

Additionally, it should be noted that the financial template design depends on a number of economic and game-theoretic arguments and assumptions. These were explored

to the extent that they clarified the intention of the code base, but we did not audit the mechanism design itself.

All external code and dependencies were assumed to work as intended.

Privileged roles

The Holdefi team currently administers all aspects of the protocol to decide which

assets can be used and how price feeds and rates are set. They also control various economic parameters, such as the size of the incentive used to encourage third parties to liquidate under-collateralized loans or which markets will have a promoted rate. All sensitive actions that owners and privileged

roles can carry out (the most sensitive ones listed below) are instant and forced, with no opt-in nor opt-out mechanism for users of the protocol.

The owner of the `Holdefi` contract can:

- Withdraw the liquidation and promotion reserves.
- Set the promotion rate.
- Set and fix the

address of the `HoldefiPrices` contract.

The owner of the contracts that inherit from the `HoldefiPauser` contract can:

- Set the pauser's address.
- Pause and unpause functionalities.

The owner of the `HoldefiPrices` contract can:

- Add stable coins.
- Set the price of an asset.

The owner of the

`HoldefiSettings` contract can:

- Set the address of the `HoldefiContract`.
- Set the rates for borrow, suppliers share, value to loan, penalty, and bonus.
- Add markets.
- Deactivate markets.
- Add collaterals.
- Deactivate collaterals.

These decisions and parameters can significantly affect the usefulness and safety of the system. It is unclear

whether a single externally owned account or a multisig account will represent these roles at the time of audit. As for now, it requires users to fully trust the Holdefi team with these privileged roles.

***Update**: While we were auditing this project, the Holdefi team found the following issue:*

The Holdefi contract makes use of the HoldefiPrices and the HoldefiSettings contracts to whitelist the assets allowed in the project. In those contracts, the intrinsic characteristics of the assets such as the price or the borrow power for a particular collateral asset are stored.

When someone has submitted some

collateral asset using the `collateralize` functions, the value is added to the account's balance by using the ERC20 `transferFrom` function while using ERC20 tokens, and with the `msg.value` value specified in the transaction for ETH.

A problem might happen if the owner of the `HoldefiPrices` contract, which is currently

the one in charge of updating the prices of the project in a centralized manner, uses the same number of digits to express the price of every asset. In that case, because different ERC20 tokens can have different number of decimals, when the `Holdefi` contract needs to perform a borrow

to a collateralized user, the outcome could be wrong.

In the borrow scenario, the current borrow power is given by:

`borrowPowerScaled = A* (balance(collateral) * price(collateral)) - ∑ balanceBorrows(market, collateral) * price(market)`

Where `A = ratesDecimal/valueToLoanRate(collateral)`.
In the same way, a new borrow would be calculated as:

`assetToBorrowValueScaled = amount(market) * price(market)`

Which only succeeds if `borrowPowerScaled > assetToBorrowValueScaled`.

This implies that a higher balance for one of the collaterals, due to a higher `decimals` value for that ERC20 token, could allow a greater borrow in real value.

Nevertheless, because all values are calculated (at

some point) by the number of tokens times its price, and because currently the oracle is updated only by its owner, the owner could compensate the decimals by adding or subtracting zeros in the price under the following formula:

`fixedPrice(market) = price(market) * 10** (18 - tokenDecimals) * priceDigits`

Where `tokenDecimals` is the decimals used for the ERC20 token and `priceDigits` is the number of digits used to scale the price.

Furthermore, because there might be ERC20 stable tokens that do not have 18 decimals, the `addStableCoin` function from the `HoldefiPrices` contract sets up wrongly a stable asset

with a hardcoded 18 decimals value into the contract.

Because the oracle is controlled by the Holdefi's owner, consider shifting the price of the assets to compensate the difference in ERC20 token decimals, and taking into account that the EIP20 Standard marks the `decimals` function as optional, allowing that

some mainnet ERC20 tokens may not have a `decimals` function to call.

Additionally, consider asking as parameter the number of decimals for non 18 decimals ERC20 stable tokens when setting up the asset into the contract.

Moreover, consider refactoring the code to include the decimals as part of the

calculation, which could allow the usage of an external oracle too.

In addition, consider explicitly documenting how the difference in decimals is handled in the project, and also consider implementing a bug bounty program.

READ ALL THE ISSUES

OpenZeppelin

Products

Contracts
Defender

Security

Security Audits

Learn

Docs
Forum
Ethernaut

Company

Website
About
Jobs
Logo Kit