

## **Summary**

Audit Report prepared by Solidified covering the AquaFi smart contracts.

# **Process and Delivery**

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief was on 15 September 2021.

The fix verification was done on 21 September 2021.

## **Audited Files**

The source code has been supplied in the form of several GitHub repositories:

https://github.com/BlockzeroLabs/aquafi-premium-contract

Commit hash: 7e4ce99da5af950310e5af4fed93d9933514e502

https://github.com/BlockzeroLabs/agua-token-contract

Commit hash: 77dced5da8073c604f01bc2ca3ea1e779c2eca17

 $\underline{https://github.com/BlockzeroLabs/aquafi-primary-smart-contract}$ 

Commit hash: d46aecfd2dfcd44717bfbfacb2ecb00c15a4d868

Commit hash for fixes: 2020552af2be5146411d2a8e67932b32b7bdb6e2

https://github.com/BlockzeroLabs/aquafi-index-fund

Commit hash: a649463823ac2e60bfa4d1058dd3ea0428889126

Commit hash for fixes: 1d3d22c09b8f8a549a22fba870c62ed63e6549c9

https://github.com/BlockzeroLabs/aquafi-uniswap-v2-handler

Commit hash: 3a09a665995baf42099ea3f221826bc3f9958522

Commit hash for fixes: 1c0f2ab55dcd2433fa18d8d3b03705647fb023c5

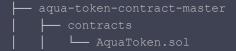
https://github.com/BlockzeroLabs/aguafi-uniswap-v3-handler

Commit hash: 3ac25072287dff693699b8d46ccfcffee1d96c42

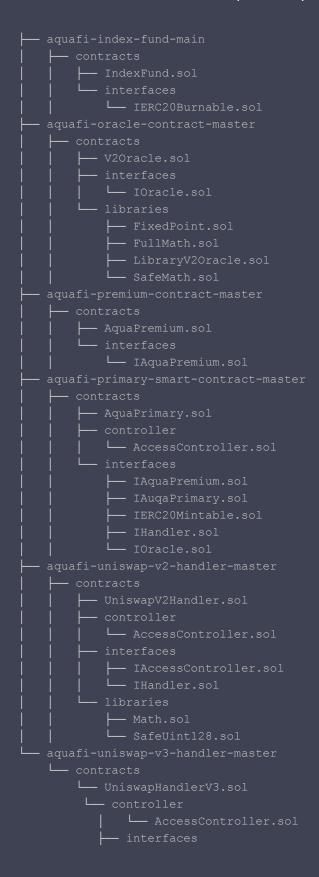
Commit hash for fixes: 29fa9d6131bbd0c277adb630896fdcc153736840

https://github.com/BlockzeroLabs/aquafi-oracle-contract

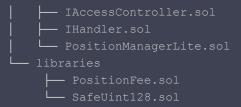
Commit hash: e49745bd13e3c225be4f3fa21780dc338931ed46











## **Intended Behavior**

The smart contracts implement a token and staking protocol.

# **Code Complexity and Test Coverage**

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	Medium	-



# **Issues Found**

Solidified found that the AquaFi contracts contain no critical issues, 1 major issue, 4 minor issues, in addition to 5 informational notes.

1 Warning aimed at end users has been noted.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	IndexFund.sol: Function withdrawWithPermission() - possible Ether loss		Resolved
2	UniswapHandlerV3.sol: Function update() - parameter pool validation is not sufficient	Minor	Resolved
3	AquaToken.sol Token susceptible to front running	Minor	Acknowledged
4	IndexFund.sol Migrated contract can cause loss of tokens	Minor	Resolved
5	Add missing zero address validations		Partially Resolved
6	Administrative accounts could drain funds if compromised	Warning	Acknowledged
7	AccessController.sol Allows whitelisting non-existent pools	Note	-
8	AquaToken.sol: Consider emitting events for all allowance updates	Note	-
9	AquaToken.sol: Token supports minting to zero address	Note	-
10	AccessController.sol: function updatePrimary() does not emit an event	Note	-
11	Miscellaneous notes	Note	-



### Critical Issues

No critical Issues found.

# **Major Issues**

# 1. IndexFund.sol: Function withdrawWithPermission() - possible Ether loss

The function withdrawWithPermission() transfer's ERC-20 tokens to the provided recipient, whereas the contract's Ether balance is transferred to the indexFundV2, without checking if the variable contract indexFundV2 is set.

#### Recommendation

Consider whether the function withdrawWithPermission() should transfer the contract's Ethereum balance to the indexFundV2. If that is the expected behaviour, consider checking if indexFundV2 is set.

**Update**: Resolved

## **Minor Issues**

# 2. UniswapHandlerV3.sol: Function update() - parameter pool validation is not sufficient

The function update() decodes the pool value from the supplied data parameter. The pool value is required to be a whitelisted pool, but the relation of the pool to the UniswapV3 positions NFT token is never validated.

As a consequence, the caller of the AquaProtocol.stake() function can specify a valid pool with lowest fees and highest premiums when staking any UniswapV3 positions NFT tokens.

#### Recommendation

Consider retrieving the pool value from the provided UniswapV3 positions NFT token. Additionally, consider removing the abi encoded pool parameter - it is mostly ignored by UniswapV2Handler.sol and it is not sufficiently validated by the UniswapHandlerV3.sol.



**Update:** Resolved

# 3. AquaToken.sol Token susceptible to front running

Changing the account allowance through the approve() method brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering. A detailed description of this vulnerability can be found here:

https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA jp-RLM

#### Recommendation

Consider mitigating this race condition by implementing increaseAllowance and decreaseAllowance functions to update the allowance.

**Update:** Acknowledged

# 4. IndexFund.sol Migrated contract can cause loss of tokens

The function burnAndWithdraw does not implement any checks to ensure whether the IndexFund contract is migrated or active. This prevents a user from accidentally calling the method and burning tokens.

This can occur for any burning that may happen after invoking the migrate and withdrawWithPermission methods.

#### Recommendation

Consider adding a validation to check whether the IndexFund is active or migrated.

**Update:** Resolved for the migrate method.

# 5. Add missing zero address validations

The contracts in several places do not validate the address in the parameter or storage. This can sometimes cause unintentional loss of funds. The following are a few such places that requires extra validation



1. IndexFund.sol: The method withdrawNftWithPermission does not validate the recipient address.

**Update**: Resolved

- 2. IndexFund.sol: Timelock can unintentionally revoke it's access by calling the updateTimelock method with a zero address.
- 3. UniswapV2Handler.sol and UniswapHandlerV3.sol: The function update() does not check if the provided staker (decoded from data parameter) is not a zero address

#### Recommendation

Consider adding the recommended validations.

# **Warnings**

# 6. Administrative accounts could drain funds if compromised

Some administrative accounts could drain funds if compromised, for example:

- 1. IndexFund.sol: the timelock account could withdraw all funds from the index.
- 2. UniswapHandlerV3.sol: the owner account could steal all users' funds from the handler by modifying the address of AQUA\_PRIMARY and updating the staker of each stake through the update() function (since no checks are done if the stake[id] already exists).
- 3. AquaPremium.sol: the timelockContract account could temporarily update the aquaPremium and mint an enormous amount of Aqua tokens.
- 4. UniswapV2Handler.sol and UniswapHandlerV3.sol: The owner account could mint Aqua tokens by changing the pool's aquaPremium or even by whitelisting some solely owned Uniswap pool designed to maximize Aqua premiums.

### Recommendation

Consider using a secure multi-signature wallet to safe-guard the administrative accounts. For item #2 consider implementing a check in <a href="UniswapHandlerV3.update">UniswapHandlerV3.update</a>() that the already existing <a href="stake[id">stake[id]</a> cannot be updated.

**Update:** Acknowledged



Response from AquaFi team - "This is known and by design - we want the upgradability once on mainnet and will finalise the protocol via governance in the near future. These functions will be controlled by the Blockzero Council multisig through the timelock controller where applicable: https://gnosis-safe.io/app/#/safes/0x5089722613C2cCEe071C39C59e9889641f435F15"

## **Notes**

# 7. AccessController.sol Allows whitelisting non-existent pools

The methods updatePremiumOfPool and updatePoolStatus in the contracts UniswapV2 AccessController and UniswapV3 AccessController allow updating the values of a non-existent pool. This results in whitelisting a pool without using the addPools method.

#### Recommendation

Consider adding a validation to check if the pool exists before updating its value.

# 8. AquaToken.sol: Consider emitting events for all allowance updates

The AquaToken contract does not emit an Approve event when the allowance value is updated during transferFrom. This does not affect the token, but any event listener that keeps track of the allowance through the events emitted will have issues synchronizing with updates.

#### Recommendation

The ERC20 standard does not mandate emitting events in this case, but it is a good practice to implement this to avoid any out-of-sync clients.

# 9. AquaToken.sol: Token supports minting to zero address

The AquaToken contract prevents the zero address from holding tokens in both burn and transfer methods. But the mint process does not implement such checks and results in address(0) having some tokens that can never be burnt or transferred.

#### Recommendation

Consider adding address(0) validation to the mint function.



# 10. AccessController.sol: updatePrimary() does not emit an event

The function updatePrimary() defined in the UniswapV3 AccessController.sol does not emit an event - it is not consistent with the UniswapV2 AccessController.updatePrimary() which does emit the AquaPrimaryUpdated event.

#### Recommendation

Consider emitting the AquaPrimaryUpdated event.

## 11. Miscellaneous notes

Miscellaneous notes for improving the code quality and readability.

- 1. IndexFund.sol: Consider adding a receive method to receive the funds instead of using a payable fallback function.
- 2. IndexFund.sol: Replace AQUA\_ADDRESS with a valid address before deploying to the main net or use the constructor to update it.
- 3. IndexFund.sol: Consider using the call method to transfer the funds in the burnAndWithdraw method since the transfer method will not always work if the target address is a smart contract with a fallback function.
- 4. AquaPremium.sol: Unused variables intitationTimestamp and intialPremium in the method calculatePremium.
- 5. UniswapV2/AccessController.sol: Unused event OwnerUpdated
- 6. UniswapV3/AccessController.sol: Consider updating the UNISWAP\_V3\_FACTORY value from the constructor.
- 7. UniswapV3/AccessController.sol: Consider adding a validation to ensure the length of tokenB and awayPremium array are the same.
- 8. IndexFund.sol: Consider checking supplied arrays are of equal length in functions withdrawWithPermission() and withdrawNftWithPermission().
- 9. AquaPrimary.sol: Consider checking supplied arrays are of equal length in the function unstake.

#### Recommendation

Consider updating the code based on the notes.



## **Disclaimer**

Solidified audit is not a security warranty, investment advice, or an endorsement of BlockZero or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.