MythX

Analysis 3a36f6e8-07a1-4dcc-a6c8-b74407081c69

MythX

Started	Fri Aug 20 2021 06:50:55 GMT+0000 (Coordinated Universal Time)
Finished	Fri Aug 20 2021 07:06:16 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Swissshares.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

/contracts/swissshares.sol

Locations

```
require(amount <= MAX_AMOUNT, "SwissShares: Maximum amount error");
require(amount % 1 == 0, "SwissShares: Can't transfer fractional amount");
if (from == address(0)) {

// Mint call
require(
isWalletWhitelisted(to),
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

/contracts/swissshares.sol

Locations

```
if (_tokenHolders[from] != 0 && _tokenHolders[from] - amount == 0) {
// Remove the wallet from token holder list
_holders.remove(from);
}
// Update the token holdings
// Update the token holdings
if (to != address(0)) _tokenHolders[to] += amount;
if (from != address(0)) _tokenHolders[from] -= amount;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

/contracts/swissshares.sol

Locations

```
}

function authTransfer(
address sender,
address recipient,
uint256 amount
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

/contracts/swissshares.sol

Locations

```
function authTransfer(
address sender,
address recipient,
uint256 amount
) internal virtual override {
_transfer(sender, recipient, amount);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/contracts/swissshares.sol

Locations

```
//SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol";
```