MythX

REPORT 600F42C4DFD70E00119EE9F0

Created	Mon Jan 25 2021 22:14:28 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	contact@rubicon.finance

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
70d4a6b3-b5bc-4730-ae2d-58c0c5362ca0	C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol	11

Analysis 70d4a6b3-b5bc-4730-ae2d-58c0c5362ca0

MythX

Started	Mon Jan 25 2021 22:14:36 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jan 25 2021 22:59:45 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	`C:\Users\Benjamin Hughes\Workspace\Rubicon\Rubicon_protocol\Contracts\Timelock.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	6	5

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setDelay" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
function() external payable {}

function setDelay(uint256 delay_) public {
require(
msg.sender == address(this),
"Timelock::setDelay: Call must come from Timelock."
);
require(
delay_ >= MINIMUM_DELAY,
"Timelock::setDelay: Delay must exceed minimum delay."
);
require(
delay_ <= MAXIMUM_DELAY,
"Timelock::setDelay: Delay must not exceed maximum delay."
);
delay = delay_;

emit NewDelay(delay);
}

function acceptAdmin() public {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "acceptAdmin" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
}

function acceptAdmin() public {
require(
msg.sender == pendingAdmin,
"Timelock::acceptAdmin: Call must come from pendingAdmin."
);
admin = msg.sender;
pendingAdmin = address(0);

emit NewAdmin(admin);
}

function setPendingAdmin(address pendingAdmin_) public {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setPendingAdmin" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
}

function setPendingAdmin(address pendingAdmin_) public {
require(
msg.sender == address(this),
"Timelock::setPendingAdmin: Call must come from Timelock."
);
pendingAdmin = pendingAdmin_;

emit NewPendingAdmin(pendingAdmin);
}

function queueTransaction(
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "queueTransaction" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
}

function queueTransaction(
address target,
uint256 value,
string memory signature,
bytes memory data,
uint256 eta
) public returns (bytes32) {
require(
msg.sender == admin,
"Timelock::queueTransaction: Call must come from admin."
);
require(
eta >= getBlockTimestamp().add(delay),
"Timelock::queueTransaction: Estimated execution block must satisfy delay."
);

bytes32 txHash =
keccak256(abi.encode(target, value, signature, data, eta));
queuedTransactions[txHash] = true;

emit QueueTransaction(txHash, target, value, signature, data, eta);
return txHash;
}

function cancelTransaction(
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "cancelTransaction" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
}

function cancelTransaction(
address target,
uint256 value,
string memory signature,
bytes memory data,
uint256 eta
) public {
require(
msg.sender == admin,
"Timelock::cancelTransaction: Call must come from admin."
);

bytes32 txHash =
keccak256(abi.encode(target, value, signature, data, eta));
queuedTransactions[txHash] = false;

emit CancelTransaction(txHash, target, value, signature, data, eta);
}

function executeTransaction(
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "executeTransaction" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
}

function executeTransaction(
address target,
uint256 value,
string memory signature,
bytes memory data,
uint256 eta
) public payable returns (bytes memory) {
require(
msg.sender == admin,
"Timelock::executeTransaction: Call must come from admin."
);

bytes32 txHash =
keccak256(abi.encode(target, value, signature, data, eta));
require(
queuedTransactions[txHash],
"Timelock::executeTransaction: Transaction hasn't been queued."
);
require(
getBlockTimestamp() >= eta,
"Timelock::executeTransaction: Transaction hasn't surpassed time lock."
);
require(
getBlockTimestamp() <= eta.add(GRACE_PERIOD),
"Timelock::executeTransaction: Transaction is stale."
);

queuedTransactions[txHash] = false;

bytes memory callData;

if (bytes(signature).length == 0) {
callData = data;
} else {
callData = abi.encodePacked(
bytes4(keccak256(bytes(signature))),
data
);
}

// solium-disable-next-line security/no-call-value
(bool success, bytes memory returnData) =
target.call.value(value)(callData);
require(
success,
"Timelock::executeTransaction: Transaction execution reverted."
);

emit ExecuteTransaction(txHash, target, value, signature, data, eta);

return returnData;
}

```

```
function getBlockTimestamp() internal view returns (uint256) {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.8"". 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

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
// File: contracts/SafeMath.sol

pragma solidity ^0.5.8;

// From https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/Math.sol
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.8"". 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

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
// File: contracts/Timelock.sol

pragma solidity ^0.5.8;

contract Timelock {
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a, "SafeMath: addition overflow");

return c;
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
"Timelock::queueTransaction: Call must come from admin."
);
require(
eta >= getBlockTimestamp().add(delay),
"Timelock::queueTransaction: Estimated execution block must satisfy delay."
);

bytes32 txHash =
```

LOW

SWC-128

Potentially unbounded data structure passed to builtin.

Gas consumption in function "executeTransaction" in contract "Timelock" depends on the size of data structures that may grow unboundedly. Specifically the "1-st" argument to builtin "keccak256" may be able to grow unboundedly causing the builtin to consume more gas than the block gas limit, effectively causing a denial-of-service condition.Consider that an attacker might attempt to cause this condition on purpose.

Source file

`C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Timelock.sol`

Locations

```
} else {
callData = abi.encodePacked(
bytes4(keccak256(bytes(signature))),
data
);
```