MythX

Analysis 75aee8d8-4593-4067-bb2c-b295fd216f19

MythX

Started	Wed Jun 02 2021 16:40:35 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jun 02 2021 16:55:46 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Timelock.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	6	2

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

/contracts/timelock.sol

Locations

```
receive() external payable { }

function setDelay(uint 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

`/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

`/contracts/timelock.sol`

Locations

```
}

function setPendingAdmin(address pendingAdmin_) public {
// allows one time setting of admin for deployment purposes
if (admin_initialized) {
require(msg.sender == address(this), "Timelock::setPendingAdmin: Call must come from Timelock.");
} else {
require(msg.sender == admin, "Timelock::setPendingAdmin: First call must come from admin.");
admin_initialized = true;
}
pendingAdmin = pendingAdmin_;

emit NewPendingAdmin(pendingAdmin);
}

function queueTransaction(address target, uint value, string memory signature, bytes memory data, uint eta) public returns (bytes32) {
```

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

`/contracts/timelock.sol`

Locations

```
}

function queueTransaction(address target, uint value, string memory signature, bytes memory data, uint 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(address target, uint value, string memory signature, bytes memory data, uint eta) public {
```

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

`/contracts/timelock.sol`

Locations

```
}

function cancelTransaction(address target, uint value, string memory signature, bytes memory data, uint 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(address target, uint value, string memory signature, bytes memory data, uint eta) public payable returns (bytes memory) {
```

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

/contracts/timelock.sol

Locations

```
}

function executeTransaction(address target, uint value, string memory signature, bytes memory data, uint 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 (uint) {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<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/timelock.sol

Locations

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

/**
```

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

`/contracts/timelock.sol`

Locations

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

```