MythX

REPORT 605D1583DD06F7001140F905

Created	Thu Mar 25 2021 22:58:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	poopswap@outlook.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
c0e2ecee-9598-439b-9cb0-1834368515c6	/contracts/timelock.sol	7

Analysis c0e2ecee-9598-439b-9cb0-1834368515c6

MythX

Started	Thu Mar 25 2021 22:58:18 GMT+0000 (Coordinated Universal Time)
Finished	Thu Mar 25 2021 23:31:23 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Timelock.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	6	1

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

```

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

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 {
// allows one time setting of admin for deployment purposes
if (admin_initialized) {
require(msg.sender == address(this), "Timelock::setPendingAdmin: Call must come from Timelock.");
```

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,
uint256 value,
string memory signature,
bytes memory data,
uint256 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

```
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(
address target,
uint256 value,
string memory signature,
bytes memory data,
uint256 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

```
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(
address target,
uint256 value,
string memory signature,
bytes memory data,
uint256 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

```
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) {
// solium-disable-next-line security/no-block-members
return block.timestamp;
}
```

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

```

// 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.");
```