MythX

REPORT 600F362AABC2930012977C5D

Created	Mon Jan 25 2021 21:20:42 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	contact@rubicon.finance

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
1bd9bfe0-d579-4c59-90fe-7e080f3e174d	C:\Users\Benjamin Hughes\workspace\rubicon\rubicon_protocol\contracts\Aqueduct.sol	10

Analysis 1bd9bfe0-d579-4c59-90fe-7e080f3e174d MythX

Started	Mon Jan 25 2021 21:20:49 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jan 25 2021 22:06:29 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\Aqueduct.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	7	3

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "delegate" 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\RBCN.sol

Locations

```
213 * @param delegatee The address to delegate votes to
214 */
215 function delegate(address delegatee) public {
216 return _delegate(msg.sender, delegatee);
217 }
218
219 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "delegateBySig" 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\RBCN.sol

Locations

```
226 * @param s Half of the ECDSA signature pair
227 */
228 function delegateBySig(
229 address delegatee,
230 uint256 nonce,
231 uint256 expiry,
232 uint8 v,
233 bytes32 r,
234 bytes32 s
235 ) public {
236 bytes32 domainSeparator =
237 keccak256(
238 abi.encode(
239 DOMAIN_TYPEHASH,
240 keccak256(bytes(name)),
241 getChainId(),
242 address(this)
243 )
244 );
245 bytes32 structHash =
246 keccak256(
247 abi.encode(DELEGATION_TYPEHASH, delegatee, nonce, expiry)
248 );
249 bytes32 digest =
250 keccak256(
251 abi.encodePacked("\x19\x01", domainSeparator, structHash)
252 );
253 address signatory = ecrecover(digest, v, r, s);
254 require(
255 signatory != address(0),
256 "RBCN::delegateBySig: invalid signature"
257 );
258 require(
259 nonce == nonces[signatory]++,
260 "RBCN::delegateBySig: invalid nonce"
261 );
262 require(now <= expiry, "RBCN::delegateBySig: signature expired");
263 return _delegate(signatory, delegatee);
264 }
265
266 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "getPriorVotes" 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\RBCN.sol

Locations

```
282 * @return The number of votes the account had as of the given block
283 */
284 function getPriorVotes(address account, uint256 blockNumber)
285 public
286 view
287 returns (uint96)
288 {
289 require(
290 blockNumber < block.number,
291 "RBCN::getPriorVotes: not yet determined"
292 );
293
294 uint32 nCheckpoints = numCheckpoints[account];
295 if (nCheckpoints == 0) {
296 return 0;
297 }
298
299 // First check most recent balance
300 if (checkpoints[account][nCheckpoints - 1].fromBlock <= blockNumber) {
301 return checkpoints[account][nCheckpoints - 1].votes;
302 }
303
304 // Next check implicit zero balance
305 if (checkpoints[account][0].fromBlock > blockNumber) {
306 return 0;
307 }
308
309 uint32 lower = 0;
310 uint32 upper = nCheckpoints - 1;
311 while (upper > lower) {
312 uint32 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
313 Checkpoint memory cp = checkpoints[account][center];
314 if (cp.fromBlock == blockNumber) {
315 return cp.votes;
316 } else if (cp.fromBlock < blockNumber) {
317 lower = center;
318 } else {
319 upper = center - 1;
320 }
321 }
322 return checkpoints[account][lower].votes;
323 }
324
325 function getDistStartTime() external view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setDistributionParams" 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\Aqueduct.sol

Locations

```
30  }
31
32  function setDistributionParams(address _RBCNAddress, address RubiconMarket)
33  public
34  onlyOwner
35  returns (bool)
36  {
37  RBCNAddress = _RBCNAddress;
38  RubiconMarketAddress = RubiconMarket;
39  RBCN = RBCNInterface(RBCNAddress);
40  timeOfLastRBCNDist = RBCN.getDistStartTime();
41  }
42
43  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setNewExchange" 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\Aqueduct.sol

Locations

```
81  * @dev Admin has the ability to choose a new exchange.
82  */
83  function setNewExchange(address newImplementation)
84  public
85  onlyOwner
86  returns (bool)
87  {
88  RubiconMarketAddress = newImplementation;
89  }
90
91  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setOwner" 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\Aqueduct.sol

Locations

```
92  * @dev Only the owner can set a new owner. Timelock can be set as owner
93  */
94  function setOwner(address newOwner) public onlyOwner returns (bool) {
95  owner = newOwner;
96  }
97
98  function getRBCNAddress() public view returns (address) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getRBCNAddress" 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\Aqueduct.sol

Locations

```
96  }
97
98  function getRBCNAddress() public view returns (address) {
99  return RBCNAddress;
100 }
101
102 // Should return the proportion of RBCN distribution per block that
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.16"". 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\Aqueduct.sol

Locations

```
1   pragma solidity ^0.5.16;
2
3   import "./RBCN.sol";
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. 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\RBCN.sol

Locations

```
288 {
289 require(
290 blockNumber < block.number,
291 "RBCN::getPriorVotes: not yet determined"
292 );
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. 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\RBCN.sol

Locations

```
420 uint32 blockNumber =
421 safe32(
422 block.number,
423 "RBCN::_writeCheckpoint: block number exceeds 32 bits"
424 );
```