MythX

REPORT 60361D809B387A0018512477

Created	Wed Feb 24 2021 09:33:52 GMT+0000 (Coordinated Universal Time)
Number of analyses	36
User	bitedama@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
1eebfe47-9eb5-4213-924f-a4417ef6a180	contracts/VoteProxy.sol	8
778bd032-7eb1-43d6-aa31-4e9133c533df	contracts/Timelock.sol	0
72572d50-00a5-40f1-aa20-0fccc546fc0f	contracts/Oracle.sol	0
3b347a91-24aa-48cf-b503-9046e472261b	contracts/ReferRank.sol	28
77456410-a578-43f6-b87a-b89d68e301c3	contracts/Boardroom.sol	6
bef376ee-8f32-43cd-9674-7c929a81bc47	Distributor.sol	3
cc487206-9172-41f7-888e-6f7f66a9fad0	contracts/Referrer.sol	2
7bb9b14f-8758-4d02-83bd-74becc4a3278	contracts/Share.sol	22
20ab3d37-9c68-4992-968b-c82b0a4ad758	contracts/SimpleERCFund.sol	10
784d8504-8fa6-4bdb-9ddc-064684b410e6	contracts/Cash.sol	22
899144ba-de21-4d7d-9088-70cf382c3f00	contracts/Bond.sol	22
d19fe2ee-4880-40b5-9a40-abb830224d10	contracts/Treasury.sol	24
5d459917-6ad8-447c-9f12-dbd351ad8507	contracts/MiningPool.sol	11
36f380f4-ca67-43a7-8941-7fd384c08137	contracts/BoardroomRank.sol	28
ae9f92b5-8429-49d6-9452-b4310b3f4eeb	contracts/token/LPTokenWrapper.sol	5
3007a760-b04f-466e-8082-c42dd1415453	contracts/token/TokenWrapper.sol	5
45f1a91a-1c99-4a17-b474-4209da71399b	contracts/token/Token.sol	15
26d6d6d9-9cff-4d24-90b3-5e0cf25ace3b	contracts/test/MockDai.sol	22
4ec286e3-6eec-42d0-8b32-de2daa179962	contracts/test/MockOracle.sol	3
2fde3f72-a42e-43b3-a39a-f50bbefa4256	contracts/test/MockBoardroom.sol	7

e26922d8-d942-467a-a3e6-5fae10d9b68f	contracts/distribution/HOCVHOTPool.sol	5
5712c1c4-a070-4a8f-8a7a-6c39a14e9ab1	contracts/distribution/HOCWHTPool.sol	5
b265e9a2-cee0-4f80-980d-8950040637fc	contracts/distribution/HOCUSDTPool.sol	5
1804a1da-7119-4f21-ab72-139b1b1a20f3	contracts/distribution/HOCUSDTLPPool.sol	5
0bf7f901-daf0-4dba-a292-621b9a88916f	contracts/distribution/HOSUSDTLPTokenSharePool.sol	5
7d93dbad-fbbe-4121-96f7-ace9f2cd5f87	contracts/distribution/HOSUSDTLPPool.sol	5
785c54a6-daf3-4f4d-8171-f8111f4972ee	contracts/distribution/HOCUSDTLPTokenSharePool.sol	5
658d8971-c09c-4bee-9f32-701f08b80a0f	contracts/distribution/VHOTSharePool.sol	5
3a2704c9-39e1-443e-9760-59635f046054	contracts/utils/Epoch.sol	11
f507a554-2819-4a1a-bb6e-841c06cdd082	contracts/utils/ContractGuard.sol	7
052b09f4-1844-4bb5-b42b-fec6cbfd0585	contracts/distributor/InitialShareDistributor.sol	21
1058ea45-7214-4200-9d4d-26d2c48334cb	FixedPoint.sol	1
8d1fccc9-23b5-4315-9d72-1f25cd6bec41	contracts/lib/Safe112.sol	1
28fe8644-5d0b-47dd-b158-222d287837a7	lib/UniswapV2OracleLibrary.sol	1
b352aa1d-b69f-4b5d-8df7-46640156d301	contracts/lib/UniswapV2Library.sol	1
6788a490-1c3d-46bf-918e-c89bf628fba6	contracts/lib/Babylonian.sol	2

Analysis 1eebfe47-9eb5-4213-924f-a4417ef6a180 MythX

Started	Wed Feb 24 2021 09:34:06 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:16 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/VoteProxy.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	7	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setBoardroom" 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/VoteProxy.sol`

Locations

```
20 }
21
22 function setBoardroom(address newBoardroom) public onlyOperator {
23 address oldBoardroom = boardroom;
24 boardroom = newBoardroom;
25 emit BoardroomChanged(msg.sender, oldBoardroom, newBoardroom);
26 }
27
28 function decimals() external pure returns (uint8) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/VoteProxy.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/IERC20.sol';
```

Analysis 778bd032-7eb1-43d6-aa31-4e9133c533df MythX

Started	Wed Feb 24 2021 09:34:16 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:23 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Timelock.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis 72572d50-00a5-40f1-aa20-0fccc546fc0f MythX

Started	Wed Feb 24 2021 09:34:16 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:34:33 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Oracle.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis 3b347a91-24aa-48cf-b503-9046e472261b MythX

Started	Wed Feb 24 2021 09:34:16 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:26 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/ReferRank.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	21	7

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
134 * - `spender` cannot be the zero address.
135 */
136 function approve(address spender, uint256 amount) public virtual override returns (bool) {
137 _approve(_msgSender(), spender, amount);
138 return true;
139 }
140
141 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
151 * `amount`.
152 */
153 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
154 _transfer(sender, recipient, amount);
155 _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
156 return true;
157 }
158
159 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
169 * - `spender` cannot be the zero address.
170 */
171 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
172 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
173 return true;
174 }
175
176 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
17 * See {ERC20-_burn}.
18 */
19 function burn(uint256 amount) public virtual {
20 _burn(_msgSender(), amount);
21 }
22
23 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
32 * `amount`.
33 */
34 function burnFrom(address account, uint256 amount) public virtual {
35 uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
36
37 _approve(account, _msgSender(), decreasedAllowance);
38 _burn(account, amount);
39 }
40 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/ReferRank.sol

Locations

```
27 * @return whether the process has been done
28 */
29 function mint(address recipient_, uint256 amount_)
30 public
31 onlyOperator
32 returns (bool)
33 {
34 uint256 balanceBefore = balanceOf(recipient_);
35 _mint(recipient_, amount_);
36 uint256 balanceAfter = balanceOf(recipient_);
37
38 return balanceAfter > balanceBefore;
39 }
40
41 function burn(uint256 amount) public override {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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/ReferRank.sol

Locations

```
39 }
40
41 function burn(uint256 amount) public override {
42 revert("can not burn");
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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/ReferRank.sol

Locations

```
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
46 revert("can not transfer");
47 }
48
49 function burnFrom(address account, uint256 amount)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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/ReferRank.sol`

Locations

```
47 }
48
49 function burnFrom(address account, uint256 amount)
50 public
51 override
52 onlyOperator
53 {
54 _burn(account, amount);
55 }
56 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/ReferRank.sol

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
* To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
*/
function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
}
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "burn" of contract "ReferRank" does not seem to be used anywhere in "burn".

Source file

`contracts/ReferRank.sol`

Locations

```
}

function burn(uint256 amount) public override {
revert("can not burn");
}
```

LOW

SWC-131

Unused function parameter "recipient".

The value of the function parameter "recipient" for the function "transfer" of contract "ReferRank" does not seem to be used anywhere in "transfer".

Source file

`contracts/ReferRank.sol`

Locations

```
}

function transfer(address recipient, uint256 amount) public override returns (bool){
revert("can not transfer");
}
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "transfer" of contract "ReferRank" does not seem to be used anywhere in "transfer".

Source file

contracts/ReferRank.sol

Locations

```
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
46 revert("can not transfer");
47 }
```

Analysis 77456410-a578-43f6-b87a-b89d68e301c3 MythX

Started	Wed Feb 24 2021 09:34:26 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:39 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Boardroom.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	5	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/Boardroom.sol

Locations

```
110
111 // stake visibility is public as overriding LPTokenWrapper's stake() function
112 function stake(uint256 amount) override public updateReward(msg.sender) {
113 require(amount > 0, "Cannot stake 0");
114 require(hasReferrer(msg.sender), "can not stake without referer");
115 super.stake(amount);
116 IMintAndBurnable(brank).mint(msg.sender, amount);
117 emit Staked(msg.sender, amount);
118 address _referer;
119 if(hasReferrer(msg.sender)){
120 _referer = referrer[msg.sender];
121 IPool(referPool).stake(_referer, amount);
122 IMintAndBurnable(rrank).mint(_referer, amount);
123 if(hasReferrer(_referer)){
124 _referer = referrer[_referer];
125 IPool(referPool).stake(_referer, amount.mul(2));
126 IMintAndBurnable(rrank).mint(_referer, amount.mul(2));
127 }
128 }
129 }
130
131 function withdraw(uint256 amount) override public updateReward(msg.sender) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setReferrer" 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/Referrer.sol`

Locations

```
15 event Refered(address indexed referrer, address indexed referree);
16
17 function setReferrer(address _referrer) public {
18 require(!hasReferrer(msg.sender), "already has referrer");
19 require(_referrer != address(0), "invalid referrer");
20 require(_referrer != msg.sender, "invalid referrer");
21 require(referreeCount(msg.sender) == 0, "already has referree");
22 referrer[msg.sender] = _referrer;
23 referree[_referrer].push(msg.sender);
24 emit Refered(_referrer, msg.sender);
25 }
26
27 function hasReferrer(address self) public view returns (bool){
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Boardroom.sol`

Locations

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "@openzeppelin/contracts/math/Math.sol";
```

Analysis bef376ee-8f32-43cd-9674-7c929a81bc47

MythX

Started	Wed Feb 24 2021 09:34:26 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:34:43 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Distributor.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "distribute" 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

`Distributor.sol`

Locations

```
10 }
11
12 function distribute() public {
13 for (uint256 i = 0; i < distributors.length; i++) {
14 distributors[i].distribute();
15 }
16 }
17 }
```

MEDIUM

SWC-128

Loop over unbounded data structure.

Gas consumption in function "distribute" in contract "Distributor" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed 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

`Distributor.sol`

Locations

```
11
12 function distribute() public {
13 for (uint256 i = 0; i < distributors.length; i++) {
14 distributors[i].distribute();
15 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

`Distributor.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  import './interfaces/IDistributor.sol';
```

Analysis cc487206-9172-41f7-888e-6f7f66a9fad0 MythX

Started	Wed Feb 24 2021 09:34:26 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:29 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Referrer.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	1	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setReferrer" 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/Referrer.sol

Locations

```
15 event Refered(address indexed referrer, address indexed referree);
16
17 function setReferrer(address _referrer) public {
18 require(!hasReferrer(msg.sender), "already has referrer");
19 require(_referrer != address(0), "invalid referrer");
20 require(_referrer != msg.sender, "invalid referrer");
21 require(referreeCount(msg.sender) == 0, "already has referree");
22 referrer[msg.sender] = _referrer;
23 referree[_referrer].push(msg.sender);
24 emit Refered(_referrer, msg.sender);
25 }
26
27 function hasReferrer(address self) public view returns (bool){
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Referrer.sol`

Locations

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

pragma solidity ^0.6.0;

import "@openzeppelin/contracts/math/SafeMath.sol";
```

Analysis 7bb9b14f-8758-4d02-83bd-74becc4a3278 MythX

Started	Wed Feb 24 2021 09:34:26 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:35 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Share.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	18	4

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount) public virtual override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* `amount`.
*/
function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
_transfer(sender, recipient, amount);
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/Share.sol`

Locations

```
16 * @param amount_ The amount of basis cash to mint to
17 */
18 function mint(address recipient_, uint256 amount_)
19 public
20 onlyOperator
21 returns (bool)
22 {
23 uint256 balanceBefore = balanceOf(recipient_);
24 _mint(recipient_, amount_);
25 uint256 balanceAfter = balanceOf(recipient_);
26 return balanceAfter >= balanceBefore;
27 }
28
29 function burn(uint256 amount) public override onlyOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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/Share.sol`

Locations

```
27 }
28
29 function burn(uint256 amount) public override onlyOperator {
30 super.burn(amount);
31 }
32
33 function burnFrom(address account, uint256 amount)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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/Share.sol`

Locations

```
31 }
32
33 function burnFrom(address account, uint256 amount)
34 public
35 override
36 onlyOperator
37 {
38 super.burnFrom(account, amount);
39 }
40 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Share.sol

Locations

```
1 pragma solidity ^0.6.0;
2
3 import './owner/Operator.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

Analysis 20ab3d37-9c68-4992-968b-c82b0a4ad758 MythX

Started	Wed Feb 24 2021 09:34:36 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:42 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/SimpleERCFund.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	8	2

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "deposit" 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/SimpleERCFund.sol

Locations

```
10 using SafeERC20 for IERC20;
11
12 function deposit(
13 address token,
14 uint256 amount,
15 string memory reason
16 ) public override {
17 IERC20(token).safeTransferFrom(msg.sender, address(this), amount);
18 emit Deposit(msg.sender, now, reason);
19 }
20
21 function withdraw(
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "withdraw" 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/SimpleERCFund.sol`

Locations

```
19  }
20
21  function withdraw(
22  address token,
23  uint256 amount,
24  address to,
25  string memory reason
26  ) public override onlyOperator {
27  IERC20(token).safeTransfer(to, amount);
28  emit Withdrawal(msg.sender, to, now, reason);
29  }
30
31  event Deposit(address indexed from, uint256 indexed at, string reason);
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17  }
18
19  function operator() public view returns (address) {
20  return _operator;
21  }
22
23  modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29  }
30
31  function isOperator() public view returns (bool) {
32  return _msgSender() == _operator;
33  }
34
35  function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/SimpleERCFund.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/IERC20.sol';
```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

`node_modules/@openzeppelin/contracts/utils/Address.sol`

Locations

```
121
122 // solhint-disable-next-line avoid-low-level-calls
123 (bool success, bytes memory returndata) = target.call{ value: weiValue }(data);
124 if (success) {
125 return returndata;
```

Analysis 784d8504-8fa6-4bdb-9ddc-064684b410e6 MythX

Started	Wed Feb 24 2021 09:34:36 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:46 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Cash.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	18	4

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
134 * - `spender` cannot be the zero address.
135 */
136 function approve(address spender, uint256 amount) public virtual override returns (bool) {
137 _approve(_msgSender(), spender, amount);
138 return true;
139 }
140
141 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
151 * `amount`.
152 */
153 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
154 _transfer(sender, recipient, amount);
155 _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
156 return true;
157 }
158
159 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
169 * - `spender` cannot be the zero address.
170 */
171 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
172 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
173 return true;
174 }
175
176 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/Cash.sol

Locations

```
28 * @return whether the process has been done
29 */
30 function mint(address recipient_, uint256 amount_)
31 public
32 onlyOperator
33 returns (bool)
34 {
35 uint256 balanceBefore = balanceOf(recipient_);
36 _mint(recipient_, amount_);
37 uint256 balanceAfter = balanceOf(recipient_);
38
39 return balanceAfter > balanceBefore;
40 }
41
42 function burn(uint256 amount) public override onlyOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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/Cash.sol`

Locations

```
40 }
41
42 function burn(uint256 amount) public override onlyOperator {
43 super.burn(amount);
44 }
45
46 function burnFrom(address account, uint256 amount)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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/Cash.sol`

Locations

```
44 }
45
46 function burnFrom(address account, uint256 amount)
47 public
48 override
49 onlyOperator
50 {
51 super.burnFrom(account, amount);
52 }
53 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Cash.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

Analysis 899144ba-de21-4d7d-9088-70cf382c3f00 MythX

Started	Wed Feb 24 2021 09:34:36 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:45 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Bond.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	18	4

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount) public virtual override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* `amount`.
*/
function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
_transfer(sender, recipient, amount);
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/Bond.sol

Locations

```
16 * @return whether the process has been done
17 */
18 function mint(address recipient_, uint256 amount_)
19 public
20 onlyOperator
21 returns (bool)
22 {
23 uint256 balanceBefore = balanceOf(recipient_);
24 _mint(recipient_, amount_);
25 uint256 balanceAfter = balanceOf(recipient_);
26
27 return balanceAfter > balanceBefore;
28 }
29
30 function burn(uint256 amount) public override onlyOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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/Bond.sol`

Locations

```
28 }
29
30 function burn(uint256 amount) public override onlyOperator {
31 super.burn(amount);
32 }
33
34 function burnFrom(address account, uint256 amount)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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/Bond.sol`

Locations

```
32 }
33
34 function burnFrom(address account, uint256 amount)
35 public
36 override
37 onlyOperator
38 {
39 super.burnFrom(account, amount);
40 }
41 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Bond.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import './owner/Operator.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

Analysis d19fe2ee-4880-40b5-9a40-abb830224d10 MythX

Started	Wed Feb 24 2021 09:34:36 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:00 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Treasury.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	16	8

ISSUES

MEDIUM **Incorrect function "_getCashPrice" state mutability**

SWC-000

Function "_getCashPrice" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file
contracts/Treasury.sol
Locations

```
131 }
132
133 function _getCashPrice(address oracle) internal view returns (uint256) {
134 try IOracle(oracle).consult(cash, 1e18) returns (uint256 price) {
135 return price;
136 } catch {
137 revert('Treasury: failed to consult cash price from the oracle');
138 }
139 }
140
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getReserve" 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/Treasury.sol

Locations

```
118
119 // budget
120 function getReserve() public view returns (uint256) {
121 return accumulatedSeigniorage;
122 }
123
124 // oracle
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getBondOraclePrice" 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/Treasury.sol

Locations

```
123
124 // oracle
125 function getBondOraclePrice() public view returns (uint256) {
126 return _getCashPrice(bondOracle);
127 }
128
129 function getSeigniorageOraclePrice() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getSeigniorageOraclePrice" 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/Treasury.sol

Locations

```
127 }
128
129 function getSeigniorageOraclePrice() public view returns (uint256) {
130 return _getCashPrice(seigniorageOracle);
131 }
132
133 function _getCashPrice(address oracle) internal view returns (uint256) {
```

MEDIUM

SWC-000

Incorrect function "estimatedCashPrice" state mutability

Function "estimatedCashPrice" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

contracts/Treasury.sol

Locations

```
140
141
142 function estimatedCashPrice() public view returns (uint256) {
143 try IOracle(seigniorageOracle).consultNow(cash, 1e18) returns (uint256 price) {
144 return price;
145 } catch {
146 revert('Treasury: failed to consult cash price from the oracle');
147 }
148 }
149
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "initialize" 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/Treasury.sol

Locations

```
152 /* ========== GOVERNANCE ========== */
153
154 function initialize() public checkOperator {
155 require(!initialized, 'Treasury: initialized');
156
157 // burn all of it's balance
158 IStableAsset(cash).burn(IERC20(cash).balanceOf(address(this)));
159
160 // set accumulatedSeigniorage to it's balance
161 accumulatedSeigniorage = IERC20(cash).balanceOf(address(this));
162
163 initialized = true;
164 emit Initialized(msg.sender, block.number);
165 }
166
167 function migrate(address target) public onlyOperator checkOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "migrate" 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/Treasury.sol`

Locations

```
165 }
166
167 function migrate(address target) public onlyOperator checkOperator {
168 require(!migrated, 'Treasury: migrated');
169
170 // cash
171 Operator(cash).transferOperator(target);
172 Operator(cash).transferOwnership(target);
173 IERC20(cash).transfer(target, IERC20(cash).balanceOf(address(this)));
174
175 // bond
176 Operator(bond).transferOperator(target);
177 Operator(bond).transferOwnership(target);
178 IERC20(bond).transfer(target, IERC20(bond).balanceOf(address(this)));
179
180 // share
181 Operator(share).transferOperator(target);
182 Operator(share).transferOwnership(target);
183 IERC20(share).transfer(target, IERC20(share).balanceOf(address(this)));
184
185 migrated = true;
186 emit Migration(target);
187 }
188
189 /* ========== MUTABLE FUNCTIONS ========== */
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getCurrentEpoch" 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/utils/Epoch.sol`

Locations

```
42 /* ========== VIEW FUNCTIONS ========== */
43
44 function getCurrentEpoch() public view returns (uint256) {
45 return epoch;
46 }
47
48 function getPeriod() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getPeriod" 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/utils/Epoch.sol`

Locations

```
46  }
47
48  function getPeriod() public view returns (uint256) {
49  return period;
50  }
51
52  function getStartTime() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getStartTime" 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/utils/Epoch.sol`

Locations

```
50  }
51
52  function getStartTime() public view returns (uint256) {
53  return startTime;
54  }
55
56  function nextEpochPoint() public view returns (uint256) {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/Treasury.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

`contracts/utils/ContractGuard.sol`

Locations

```
5
6  function checkSameOriginReentranted() internal view returns (bool) {
7  return _status[block.number][tx.origin];
8  }
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

`contracts/utils/ContractGuard.sol`

Locations

```
24 _;
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
```

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

`contracts/Treasury.sol`

Locations

```
162
163 initialized = true;
164 emit Initialized(msg.sender, block.number);
165 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
5
6  function checkSameOriginReentranted() internal view returns (bool) {
7  return _status[block.number][tx.origin];
8  }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
9
10 function checkSameSenderReentranted() internal view returns (bool) {
11 return _status[block.number][msg.sender];
12 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
24 _;
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
29 }
```

Analysis 5d459917-6ad8-447c-9f12-dbd351ad8507 MythX

Started	Wed Feb 24 2021 09:34:46 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:55 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/MiningPool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	10	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/MiningPool.sol`

Locations

```
94
95 // stake visibility is public as overriding LPTokenWrapper's stake() function
96 function stake(address account, uint256 amount) override public updateReward(msg.sender) onlyOperator {
97 require(amount > 0, "Cannot stake 0");
98 super.stake(account, amount);
99 emit Staked(account, amount);
100 }
101
102 function withdraw(address account, uint256 amount) override public updateReward(msg.sender) onlyOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "withdraw" 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/MiningPool.sol`

Locations

```
100 }
101
102 function withdraw(address account, uint256 amount) override public updateReward(msg.sender) onlyOperator {
103 require(amount > 0, "Cannot withdraw 0");
104 super.withdraw(account, amount);
105 emit Withdrawn(account, amount);
106 }
107
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getReward" 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/MiningPool.sol

Locations

```
107
108
109 function getReward() public updateReward(msg.sender) {
110 uint256 reward = earned(msg.sender);
111 if (reward > 0) {
112 rewards[msg.sender] = 0;
113 target.safeTransfer(msg.sender, reward);
114 emit RewardPaid(msg.sender, reward);
115 }
116 }
117
118 function notifyRewardAmount(uint256 reward)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setReferrer" 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/Referrer.sol

Locations

```
15 event Refered(address indexed referrer, address indexed referree);
16
17 function setReferrer(address _referrer) public {
18 require(!hasReferrer(msg.sender), "already has referrer");
19 require(_referrer != address(0), "invalid referrer");
20 require(_referrer != msg.sender, "invalid referrer");
21 require(referreeCount(msg.sender) == 0, "already has referree");
22 referrer[msg.sender] = _referrer;
23 referree[_referrer].push(msg.sender);
24 emit Refered(_referrer, msg.sender);
25 }
26
27 function hasReferrer(address self) public view returns (bool){
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/MiningPool.sol`

Locations

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

pragma solidity ^0.6.0;

import "@openzeppelin/contracts/math/Math.sol";
```

Analysis 36f380f4-ca67-43a7-8941-7fd384c08137 MythX

Started	Wed Feb 24 2021 09:34:46 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:56 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/BoardroomRank.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	21	7

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount) public virtual override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* `amount`.
*/
function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
_transfer(sender, recipient, amount);
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
17 * See {ERC20-_burn}.
18 */
19 function burn(uint256 amount) public virtual {
20 _burn(_msgSender(), amount);
21 }
22
23 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
32 * `amount`.
33 */
34 function burnFrom(address account, uint256 amount) public virtual {
35 uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
36
37 _approve(account, _msgSender(), decreasedAllowance);
38 _burn(account, amount);
39 }
40 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/BoardroomRank.sol

Locations

```
27 * @return whether the process has been done
28 */
29 function mint(address recipient_, uint256 amount_)
30 public
31 onlyOperator
32 returns (bool)
33 {
34 uint256 balanceBefore = balanceOf(recipient_);
35 _mint(recipient_, amount_);
36 uint256 balanceAfter = balanceOf(recipient_);
37
38 return balanceAfter > balanceBefore;
39 }
40
41 function burn(uint256 amount) public override {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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/BoardroomRank.sol

Locations

```
39 }
40
41 function burn(uint256 amount) public override {
42 revert("can not burn");
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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/BoardroomRank.sol

Locations

```
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
46 revert("can not transfer");
47 }
48
49 function burnFrom(address account, uint256 amount)
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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/BoardroomRank.sol`

Locations

```
47 }
48
49 function burnFrom(address account, uint256 amount)
50 public
51 override
52 onlyOperator
53 {
54 _burn(account, amount);
55 }
56 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/BoardroomRank.sol

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
* To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
*/
function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
}
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "burn" of contract "BoardroomRank" does not seem to be used anywhere in "burn".

Source file

`contracts/BoardroomRank.sol`

Locations

```
}

function burn(uint256 amount) public override {
revert("can not burn");
}
```

LOW

SWC-131

Unused function parameter "recipient".

The value of the function parameter "recipient" for the function "transfer" of contract "BoardroomRank" does not seem to be used anywhere in "transfer".

Source file

`contracts/BoardroomRank.sol`

Locations

```
}

function transfer(address recipient, uint256 amount) public override returns (bool){
revert("can not transfer");
}
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "transfer" of contract "BoardroomRank" does not seem to be used anywhere in "transfer".

Source file

contracts/BoardroomRank.sol

Locations

```
43 }
44
45 function transfer(address recipient, uint256 amount) public override returns (bool){
46 revert("can not transfer");
47 }
```

Analysis ae9f92b5-8429-49d6-9452-b4310b3f4eeb MythX

Started	Wed Feb 24 2021 09:34:46 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:51 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Token/LPTokenWrapper.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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/token/LPTokenWrapper.sol`

Locations

```
14 mapping(address => uint256) private _balances;
15
16 function totalSupply() public view returns (uint256) {
17 return _totalSupply;
18 }
19
20 function balanceOf(address account) public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "balanceOf" 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/token/LPTokenWrapper.sol`

Locations

```
18 }
19
20 function balanceOf(address account) public view returns (uint256) {
21 return _balances[account];
22 }
23
24 function stake(uint256 amount) public virtual {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/token/LPTokenWrapper.sol`

Locations

```
22 }
23
24 function stake(uint256 amount) public virtual {
25 _totalSupply = _totalSupply.add(amount);
26 _balances[msg.sender] = _balances[msg.sender].add(amount);
27 lpt.safeTransferFrom(msg.sender, address(this), amount);
28 }
29
30 function withdraw(uint256 amount) public virtual {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "withdraw" 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/token/LPTokenWrapper.sol`

Locations

```
28 }
29
30 function withdraw(uint256 amount) public virtual {
31 _totalSupply = _totalSupply.sub(amount);
32 _balances[msg.sender] = _balances[msg.sender].sub(amount);
33 lpt.safeTransfer(msg.sender, amount);
34 }
35 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/token/LPTokenWrapper.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/math/SafeMath.sol';
```

Analysis 3007a760-b04f-466e-8082-c42dd1415453 MythX

Started	Wed Feb 24 2021 09:34:56 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:59 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Token/TokenWrapper.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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/token/TokenWrapper.sol`

Locations

```
14 mapping(address => uint256) private _balances;
15
16 function totalSupply() public view returns (uint256) {
17 return _totalSupply;
18 }
19
20 function balanceOf(address account) public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "balanceOf" 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/token/TokenWrapper.sol`

Locations

```
18 }
19
20 function balanceOf(address account) public view returns (uint256) {
21 return _balances[account];
22 }
23
24 function stake(uint256 amount) public virtual {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/token/TokenWrapper.sol`

Locations

```
22 }
23
24 function stake(uint256 amount) public virtual {
25 _totalSupply = _totalSupply.add(amount);
26 _balances[msg.sender] = _balances[msg.sender].add(amount);
27 token.safeTransferFrom(msg.sender, address(this), amount);
28 }
29
30 function withdraw(uint256 amount) public virtual {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "withdraw" 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/token/TokenWrapper.sol`

Locations

```
28 }
29
30 function withdraw(uint256 amount) public virtual {
31 _totalSupply = _totalSupply.sub(amount);
32 _balances[msg.sender] = _balances[msg.sender].sub(amount);
33 token.safeTransfer(msg.sender, amount);
34 }
35 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/token/TokenWrapper.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/math/SafeMath.sol';
```

Analysis 45f1a91a-1c99-4a17-b474-4209da71399b MythX

Started	Wed Feb 24 2021 09:34:56 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:03 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Token/Token.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	11	4

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
64  * @dev Returns the name of the token.
65  */
66  function name() public view returns (string memory) {
67  return _name;
68  }
69
70  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
72  * name.
73  */
74  function symbol() public view returns (string memory) {
75  return _symbol;
76  }
77
78  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "balanceOf" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
103 * @dev See {IERC20-balanceOf}.
104 */
105 function balanceOf(address account) public view override returns (uint256) {
106 return _balances[account];
107 }
108
109 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "allowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
123 * @dev See {IERC20-allowance}.
124 */
125 function allowance(address owner, address spender) public view virtual override returns (uint256) {
126 return _allowances[owner][spender];
127 }
128
129 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
134 * - `spender` cannot be the zero address.
135 */
136 function approve(address spender, uint256 amount) public virtual override returns (bool) {
137 _approve(_msgSender(), spender, amount);
138 return true;
139 }
140
141 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
151 * `amount`.
152 */
153 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
154 _transfer(sender, recipient, amount);
155 _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
156 return true;
157 }
158
159 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
169 * - `spender` cannot be the zero address.
170 */
171 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
172 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
173 return true;
174 }
175
176 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

LOW

SWC-103

No pragma is set.

It is recommended to make a conscious choice on what version of Solidity is used for compilation. Currently no version is set in the Solidity file.

Source file

`contracts/token/Token.sol`

Locations

```
import '@openzeppelin/contracts/token/ERC20/ERC20.sol';import '@openzeppelin/contracts/token/ERC20/ERC20.sol';

contract Token is ERC20 {
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

Analysis 26d6d6d9-9cff-4d24-90b3-5e0cf25ace3b MythX

Started	Wed Feb 24 2021 09:35:06 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:16 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Test/MockDai.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	18	4

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
64 * @dev Returns the name of the token.
65 */
66 function name() public view returns (string memory) {
67 return _name;
68 }
69
70 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
72 * name.
73 */
74 function symbol() public view returns (string memory) {
75 return _symbol;
76 }
77
78 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
89  * {IERC20-balanceOf} and {IERC20-transfer}.
90  */
91  function decimals() public view returns (uint8) {
92  return _decimals;
93  }
94
95  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
96  * @dev See {IERC20-totalSupply}.
97  */
98  function totalSupply() public view override returns (uint256) {
99  return _totalSupply;
100 }
101
102 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
115 * - the caller must have a balance of at least `amount`.
116 */
117 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
118 _transfer(_msgSender(), recipient, amount);
119 return true;
120 }
121
122 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
134 * - `spender` cannot be the zero address.
135 */
136 function approve(address spender, uint256 amount) public virtual override returns (bool) {
137 _approve(_msgSender(), spender, amount);
138 return true;
139 }
140
141 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
151 * `amount`.
152 */
153 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
154 _transfer(sender, recipient, amount);
155 _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
156 return true;
157 }
158
159 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
169 * - `spender` cannot be the zero address.
170 */
171 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
172 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
173 return true;
174 }
175
176 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
192 return true;
193 }
194
195 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
17 * See {ERC20-_burn}.
18 */
19 function burn(uint256 amount) public virtual {
20 _burn(_msgSender(), amount);
21 }
22
23 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" 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

node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol

Locations

```
32 * `amount`.
33 */
34 function burnFrom(address account, uint256 amount) public virtual {
35 uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
36
37 _approve(account, _msgSender(), decreasedAllowance);
38 _burn(account, amount);
39 }
40 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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/test/MockDai.sol

Locations

```
21  * @return whether the process has been done
22  */
23  function mint(address recipient_, uint256 amount_)
24  public
25  onlyOperator
26  returns (bool)
27  {
28  uint256 balanceBefore = balanceOf(recipient_);
29  _mint(recipient_, amount_);
30  uint256 balanceAfter = balanceOf(recipient_);
31
32  return balanceAfter > balanceBefore;
33  }
34  }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/test/MockDai.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol';
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304  * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305  */
306  function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307  }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
304 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
305 */
306 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
307 }
```

Analysis 4ec286e3-6eec-42d0-8b32-de2daa179962 MythX

Started	Wed Feb 24 2021 09:35:06 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:10 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Test/MockOracle.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setPrice" 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/test/MockOracle.sol`

Locations

```
11 bool public error;
12
13 function setPrice(uint256 _price) public {
14 price = _price;
15 }
16
17 function setRevert(bool _error) public {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setRevert" 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/test/MockOracle.sol`

Locations

```
15 }
16
17 function setRevert(bool _error) public {
18 error = _error;
19 }
20
21 function update() external override {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/test/MockOracle.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/SafeMath.sol';
```

Analysis 2fde3f72-a42e-43b3-a39a-f50bbefa4256 MythX

Started	Wed Feb 24 2021 09:35:06 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:11 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Test/MockBoardroom.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	6	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/test/MockBoardroom.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/token/ERC20/IERC20.sol';
```

Analysis e26922d8-d942-467a-a3e6-5fae10d9b68f MythX

Started	Wed Feb 24 2021 09:35:16 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:26 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/HOCVHOTPool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOCVHOTPool.sol

Locations

```
80
81 // stake visibility is public as overriding LPTokenWrapper's stake() function
82 function stake(uint256 amount)
83 public
84 override
85 updateReward(msg.sender)
86 checkStart
87 {
88 require(amount > 0, 'Pool: Cannot stake 0');
89 uint256 newDeposit = deposits[msg.sender].add(amount);
90 require(
91 newDeposit <= 20000e18,
92 'Pool: deposit amount exceeds maximum 20000'
93 );
94 deposits[msg.sender] = newDeposit;
95 super.stake(amount);
96 emit Staked(msg.sender, amount);
97 }
98
99 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOCVHOTPool.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

Analysis 5712c1c4-a070-4a8f-8a7a-6c39a14e9ab1 MythX

Started	Wed Feb 24 2021 09:35:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:25 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Distribution/HOCWHTPool.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOCWHTPool.sol

Locations

```
80
81 // stake visibility is public as overriding LPTokenWrapper's stake() function
82 function stake(uint256 amount)
83 public
84 override
85 updateReward(msg.sender)
86 checkStart
87 {
88 require(amount > 0, 'Pool: Cannot stake 0');
89 uint256 newDeposit = deposits[msg.sender].add(amount);
90 require(
91 newDeposit <= 20000e18,
92 'Pool: deposit amount exceeds maximum 20000'
93 );
94 deposits[msg.sender] = newDeposit;
95 super.stake(amount);
96 emit Staked(msg.sender, amount);
97 }
98
99 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOCWHTPool.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

Analysis b265e9a2-cee0-4f80-980d-8950040637fc MythX

Started	Wed Feb 24 2021 09:35:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:26 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/HOCUSDTPool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOCUSDTPool.sol

Locations

```
80
81 // stake visibility is public as overriding LPTokenWrapper's stake() function
82 function stake(uint256 amount)
83 public
84 override
85 updateReward(msg.sender)
86 checkStart
87 {
88 require(amount > 0, 'Pool: Cannot stake 0');
89 uint256 newDeposit = deposits[msg.sender].add(amount);
90 require(
91 newDeposit <= 20000e18,
92 'Pool: deposit amount exceeds maximum 20000'
93 );
94 deposits[msg.sender] = newDeposit;
95 super.stake(amount);
96 emit Staked(msg.sender, amount);
97 }
98
99 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOCUSDTPool.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

Analysis 1804a1da-7119-4f21-ab72-139b1b1a20f3 MythX

Started	Wed Feb 24 2021 09:35:27 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:38 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/HOCUSDTLPPool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOCUSDTLPPool.sol

Locations

```
80
81 // stake visibility is public as overriding LPTokenWrapper's stake() function
82 function stake(uint256 amount)
83 public
84 override
85 updateReward(msg.sender)
86 checkStart
87 {
88 require(amount > 0, 'Pool: Cannot stake 0');
89 uint256 newDeposit = deposits[msg.sender].add(amount);
90 require(
91 newDeposit <= 20000e18,
92 'Pool: deposit amount exceeds maximum 20000'
93 );
94 deposits[msg.sender] = newDeposit;
95 super.stake(amount);
96 emit Staked(msg.sender, amount);
97 }
98
99 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOCUSDTLPPool.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

Analysis `0bf7f901-daf0-4dba-a292-621b9a88916f` MythX

Started	Wed Feb 24 2021 09:35:27 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:34 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Distribution/HOSUSDTLPTokenSharePool.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOSUSDTLPTokenSharePool.sol

Locations

```
140
141 // stake visibility is public as overriding LPTokenWrapper's stake() function
142 function stake(uint256 amount)
143 public
144 override
145 updateReward(msg.sender)
146 checkStart
147 {
148 require(amount > 0, 'DAIBASLPTokenSharePool: Cannot stake 0');
149 super.stake(amount);
150 emit Staked(msg.sender, amount);
151 }
152
153 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOSUSDTLPTokenSharePool.sol

Locations

```
1 pragma solidity ^0.6.0;
2 /**
3 *Submitted for verification at Etherscan.io on 2020-07-17
```

Analysis 7d93dbad-fbbe-4121-96f7-ace9f2cd5f87 MythX

Started	Wed Feb 24 2021 09:35:27 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:36 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/HOSUSDTLPPool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOSUSDTLPPool.sol

Locations

```
80
81 // stake visibility is public as overriding LPTokenWrapper's stake() function
82 function stake(uint256 amount)
83 public
84 override
85 updateReward(msg.sender)
86 checkStart
87 {
88 require(amount > 0, 'Pool: Cannot stake 0');
89 uint256 newDeposit = deposits[msg.sender].add(amount);
90 require(
91 newDeposit <= 20000e18,
92 'Pool: deposit amount exceeds maximum 20000'
93 );
94 deposits[msg.sender] = newDeposit;
95 super.stake(amount);
96 emit Staked(msg.sender, amount);
97 }
98
99 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOSUSDTLPPool.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/Math.sol';
```

Analysis 785c54a6-daf3-4f4d-8171-f8111f4972ee MythX

Started	Wed Feb 24 2021 09:35:37 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:45 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/HOCUSDTLPTokenSharePool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/HOCUSDTLPTokenSharePool.sol

Locations

```
133
134 // stake visibility is public as overriding LPTokenWrapper's stake() function
135 function stake(uint256 amount)
136 public
137 override
138 updateReward(msg.sender)
139 checkhalve
140 checkStart
141 {
142 require(amount > 0, 'Cannot stake 0');
143 super.stake(amount);
144 emit Staked(msg.sender, amount);
145 }
146
147 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/HOCUSDTLPTokenSharePool.sol

Locations

```
1 pragma solidity ^0.6.0;
2 /**
3 *Submitted for verification at Etherscan.io on 2020-07-17
```

Analysis 658d8971-c09c-4bee-9f32-701f08b80a0f MythX

Started	Wed Feb 24 2021 09:35:37 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:47 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distribution/VHOTSharePool.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	4	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "stake" 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/distribution/VHOTSharePool.sol

Locations

```
140
141 // stake visibility is public as overriding LPTokenWrapper's stake() function
142 function stake(uint256 amount)
143 public
144 override
145 updateReward(msg.sender)
146 checkStart
147 {
148 require(amount > 0, 'DAIBASLPTokenSharePool: Cannot stake 0');
149 super.stake(amount);
150 emit Staked(msg.sender, amount);
151 }
152
153 function withdraw(uint256 amount)
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distribution/VHOTSharePool.sol

Locations

```
1 pragma solidity ^0.6.0;
2 /**
3 *Submitted for verification at Etherscan.io on 2020-07-17
```

Analysis 3a2704c9-39e1-443e-9760-59635f046054 MythX

Started	Wed Feb 24 2021 09:35:37 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:37:43 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Utils/Epoch.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	10	1

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "operator" 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/owner/Operator.sol`

Locations

```
17 }
18
19 function operator() public view returns (address) {
20 return _operator;
21 }
22
23 modifier onlyOperator() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isOperator" 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/owner/Operator.sol`

Locations

```
29 }
30
31 function isOperator() public view returns (bool) {
32 return _msgSender() == _operator;
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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/owner/Operator.sol`

Locations

```
33 }
34
35 function transferOperator(address newOperator_) public onlyOwner {
36 _transferOperator(newOperator_);
37 }
38
39 function _transferOperator(address newOperator_) internal {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getCurrentEpoch" 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/utils/Epoch.sol`

Locations

```
42 /* ========== VIEW FUNCTIONS ========== */
43
44 function getCurrentEpoch() public view returns (uint256) {
45 return epoch;
46 }
47
48 function getPeriod() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getPeriod" 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/utils/Epoch.sol`

Locations

```
46 }
47
48 function getPeriod() public view returns (uint256) {
49 return period;
50 }
51
52 function getStartTime() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getStartTime" 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/utils/Epoch.sol`

Locations

```
50 }
51
52 function getStartTime() public view returns (uint256) {
53 return startTime;
54 }
55
56 function nextEpochPoint() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "nextEpochPoint" 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/utils/Epoch.sol`

Locations

```
54 }
55
56 function nextEpochPoint() public view returns (uint256) {
57 return startTime.add(epoch.mul(period));
58 }
59
60 /* ========== GOVERNANCE ========== */
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/utils/Epoch.sol`

Locations

```
1 pragma solidity ^0.6.0;
2
3 import '@openzeppelin/contracts/math/SafeMath.sol';
```

Analysis f507a554-2819-4a1a-bb6e-841c06cdd082 MythX

Started	Wed Feb 24 2021 09:35:37 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:03 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Utils/ContractGuard.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	7

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". 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/utils/ContractGuard.sol`

Locations

```
1  pragma solidity ^0.6.12;
2
3  contract ContractGuard {
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

`contracts/utils/ContractGuard.sol`

Locations

```
5
6  function checkSameOriginReentranted() internal view returns (bool) {
7  return _status[block.number][tx.origin];
8  }
```

LOW

SWC-115

Use of "tx.origin" as a part of authorization control.

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source file

`contracts/utils/ContractGuard.sol`

Locations

```
24 _;
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
5
6 function checkSameOriginReentranted() internal view returns (bool) {
7 return _status[block.number][tx.origin];
8 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
9
10 function checkSameSenderReentranted() internal view returns (bool) {
11 return _status[block.number][msg.sender];
12 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
24 _;
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
```

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

`contracts/utils/ContractGuard.sol`

Locations

```
25
26 _status[block.number][tx.origin] = true;
27 _status[block.number][msg.sender] = true;
28 }
29 }
```

Analysis 052b09f4-1844-4bb5-b42b-fec6cbfd0585 MythX

Started	Wed Feb 24 2021 09:35:47 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:38:02 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Distributor/InitialShareDistributor.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	18	3

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
33 * @dev Returns the address of the current owner.
34 */
35 function owner() public view returns (address) {
36 return _owner;
37 }
38
39 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "renounceOwnership" 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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52 * thereby removing any functionality that is only available to the owner.
53 */
54 function renounceOwnership() public virtual onlyOwner {
55 emit OwnershipTransferred(_owner, address(0));
56 _owner = address(0);
57 }
58
59 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" 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

`node_modules/@openzeppelin/contracts/access/Ownable.sol`

Locations

```
61 * Can only be called by the current owner.
62 */
63 function transferOwnership(address newOwner) public virtual onlyOwner {
64 require(newOwner != address(0), "Ownable: new owner is the zero address");
65 emit OwnershipTransferred(_owner, newOwner);
66 _owner = newOwner;
67 }
68 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "distribute" 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/distributor/InitialShareDistributor.sol`

Locations

```
34 }
35
36 function distribute() public override {
37 require(
38 once,
39 'InitialShareDistributor: you cannot run this function twice'
40 );
41
42 share.transfer(address(daibacLPPool), daibacInitialBalance);
43 daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44 emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46 share.transfer(address(daibasLPPool), daibasInitialBalance);
47 daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48 emit Distributed(address(daibasLPPool), daibasInitialBalance);
49
50 once = false;
51 }
52 }
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
41
42  share.transfer(address(daibacLPPool), daibacInitialBalance);
43  daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
```

MEDIUM

SWC-107

Write to persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
41
42  share.transfer(address(daibacLPPool), daibacInitialBalance);
43  daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
42  share.transfer(address(daibacLPPool), daibacInitialBalance);
43  daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
42  share.transfer(address(daibacLPPool), daibacInitialBalance);
43  daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Write to persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
44  emit Distributed(address(daibacLPPool), daibacInitialBalance);
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Write to persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
45
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
49
50  once = false;
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
46  share.transfer(address(daibasLPPool), daibasInitialBalance);
47  daibasLPPool.notifyRewardAmount(daibasInitialBalance);
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
49
50  once = false;
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
48  emit Distributed(address(daibasLPPool), daibasInitialBalance);
49
50  once = false;
51  }
52  }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/distributor/InitialShareDistributor.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/SafeMath.sol';
```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
40 );
41
42 share.transfer(address(daibacLPPool), daibacInitialBalance);
43 daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44 emit Distributed(address(daibacLPPool), daibacInitialBalance);
```

LOW

SWC-113

Multiple calls are executed in the same transaction.

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

Source file

`contracts/distributor/InitialShareDistributor.sol`

Locations

```
41
42 share.transfer(address(daibacLPPool), daibacInitialBalance);
43 daibacLPPool.notifyRewardAmount(daibacInitialBalance);
44 emit Distributed(address(daibacLPPool), daibacInitialBalance);
```

Analysis 1058ea45-7214-4200-9d4d-26d2c48334cb MythX

Started	Wed Feb 24 2021 09:35:47 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:21 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	FixedPoint.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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
FixedPoint.sol
Locations

```
1  pragma solidity ^0.6.0;
2
3  import './Babylonian.sol';
```

Analysis 8d1fccc9-23b5-4315-9d72-1f25cd6bec41 MythX

Started	Wed Feb 24 2021 09:35:47 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:18 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Contracts/Lib/Safe112.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/lib/Safe112.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  library Safe112 {
```

Analysis 28fe8644-5d0b-47dd-b158-222d287837a7 MythX

Started	Wed Feb 24 2021 09:36:07 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:40 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	`Lib/UniswapV2OracleLibrary.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

`lib/UniswapV2OracleLibrary.sol`

Locations

```
1  pragma solidity ^0.6.0;
2
3  import './FixedPoint.sol';
```

Analysis b352aa1d-b69f-4b5d-8df7-46640156d301 MythX

Started	Wed Feb 24 2021 09:36:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:46 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Lib/UniswapV2Library.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/lib/UniswapV2Library.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  import '@openzeppelin/contracts/math/SafeMath.sol';
```

Analysis 6788a490-1c3d-46bf-918e-c89bf628fba6 MythX

Started	Wed Feb 24 2021 09:36:27 GMT+0000 (Coordinated Universal Time)
Finished	Wed Feb 24 2021 09:36:54 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/Lib/Babylonian.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/lib/Babylonian.sol

Locations

```
1  pragma solidity ^0.6.0;
2
3  library Babylonian {
```

LOW

SWC-128

Loop over unbounded data structure.

Gas consumption in function "sqrt" in contract "Babylonian" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed 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/lib/Babylonian.sol

Locations

```
6  z = y;
7  uint256 x = y / 2 + 1;
8  while (x < z) {
9  z = x;
10 x = (y / x + x) / 2;
```