Layer2DAO

Smart Contract Audit Report

SYSFIXED

30 Jan 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Layer2DAO	L2DAO	Arbitrum

Addresses

Contract address	0x2cab3abfc1670d1a452df502e216a66883cdf079
Contract deployer address	0xFd0Bd19e849493F77D8f77eD026520C1368102Bd

Project Website

https://www.layer2dao.org/

Codebase

https://arbiscan.io/address/0x2cab3abfc1670d1a452df502e216a66883cdf079#code

SUMMARY

Layer2DAO is expanding the Ethereum L2 ecosystem and investing in L2 ecosystem projects. The DAO is using its treasury to support high-impact L2 protocols and ecosystem plays, serving as a diversified venture fund for investors seeking exposure to the L2 ecosystem growth. It also provides liquidity, depositing and staking, perpetually reinvesting proceeds into the DAO.

Contract Summary

Documentation Quality

Layer2DAO provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Layer2DAO with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 47, 336, 336, 347, 347, 347, 380, 388, 423, 424, 428, 429, 429, 430, 445, 455, 455, 458, 458, 458, 1252, 1271, 1293, 1326, 1328, 1349, 1350, 1375, 1377, 1616, 1666, 1670, 1782, 1785, 1786, 1793, 1797, 1851, 1853, 1853, 1853, 1853, 1853, 1864, 1879, 1913, 1913, 388, 1616, 1670, 1782, 1785 and 1786.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11, 66, 310, 356, 402, 472, 708, 814, 877, 904, 982, 1067, 1097, 1455, 1544, 1804 and 1888.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 48, 429, 456, 457, 459, 459, 1594, 1616, 1663, 1670, 1782, 1785 and 1786.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 1627, 1640, 1785 and 1788.

CONCLUSION

We have audited the Layer2DAO project released in January 2022 to discover issues and identify potential security vulnerabilities in Layer2DAO Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides satisfactory results with low-risk issues.

The issues found in the Layer2DAO smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, and out-of-bounds array access which the index access expression can cause an exception in case an invalid array index value is used.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Saturday Jan 29 2022 10:28:36 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Jan 30 2022 17:44:12 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	L2DAOToken.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/=" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "--" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged

SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 47

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
46   bytes32 computedHash = leaf;
47   for (uint256 i = 0; i < proof.length; i++) {
48   bytes32 proofElement = proof[i];
49   if (computedHash <= proofElement) {
50   // Hash(current computed hash + current element of the proof)
51
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 336

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
335   // (a + b) / 2 can overflow.
336   return (a & b) + (a ^ b) / 2;
337   }
338
339   /**
340
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 336

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
335   // (a + b) / 2 can overflow.
336   return (a & b) + (a ^ b) / 2;
337   }
338
339   /**
340
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 347

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
346   // (a + b - 1) / b can overflow on addition, so we distribute.
347   return a / b + (a % b == 0 ? 0 : 1);
348   }
349   }
350
351
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 347

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
346   // (a + b - 1) / b can overflow on addition, so we distribute.
347   return a / b + (a % b == 0 ? 0 : 1);
348   }
349   }
350
351
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 347

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
346   // (a + b - 1) / b can overflow on addition, so we distribute.
347   return a / b + (a % b == 0 ? 0 : 1);
348   }
349   }
350
351
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 380

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
379   unchecked {
380   counter._value += 1;
381   }
382   }
383
384
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 388

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
387   unchecked {
388   counter._value = value - 1;
389   }
390   }
391
392
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 423

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
422   while (temp != 0) {
423   digits++;
424   temp /= 10;
425   }
426   bytes memory buffer = new bytes(digits);
427
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 424

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
423   digits++;
424   temp /= 10;
425   }
426   bytes memory buffer = new bytes(digits);
427   while (value != 0) {
428
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 428

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
427   while (value != 0) {
428   digits -= 1;
429   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
430   value /= 10;
431   }
432
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 429

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
428   digits -= 1;
429   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
430   value /= 10;
431   }
432   return string(buffer);
433
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 429

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
428   digits -= 1;
429   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
430   value /= 10;
431   }
432   return string(buffer);
433
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 430

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
429   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
430   value /= 10;
431   }
432   return string(buffer);
433   }
434
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 445

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
444   while (temp != 0) {
445   length++;
446   temp >>= 8;
447   }
448   return toHexString(value, length);
449
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 455

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
454   function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
455   bytes memory buffer = new bytes(2 * length + 2);
456   buffer[0] = "0";
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 455

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
454   function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
455   bytes memory buffer = new bytes(2 * length + 2);
456   buffer[0] = "0";
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 458

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461   }
462
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 458

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461   }
462
```

SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 458

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461   }
462
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1252

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1251   unchecked {
1252   _approve(sender, _msgSender(), currentAllowance - amount);
1253   }
1254
1255   return true;
1256
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1271

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1270   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
1271   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
1272   return true;
1273   }
1274
1275
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1293

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1292   unchecked {
1293   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
1294   }
1295
1296   return true;
1297
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1326

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1325   unchecked {
1326   _balances[sender] = senderBalance - amount;
1327   }
1328   _balances[recipient] += amount;
1329
1330
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1328

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1327   }
1328   _balances[recipient] += amount;
1329
1330   emit Transfer(sender, recipient, amount);
1331
1332
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1349

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1348
1349   _totalSupply += amount;
1350   _balances[account] += amount;
1351   emit Transfer(address(0), account, amount);
1352
1353
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1350

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1349   _totalSupply += amount;
1350   _balances[account] += amount;
1351   emit Transfer(address(0), account, amount);
1352
1353   _afterTokenTransfer(address(0), account, amount);
1354
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1375

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1374   unchecked {
1375   _balances[account] = accountBalance - amount;
1376   }
1377   _totalSupply -= amount;
1378
1379
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1377

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1376   }
1377   _totalSupply -= amount;
1378
1379   emit Transfer(account, address(0), amount);
1380
1381
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1616

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1615   uint256 pos = _checkpoints[account].length;
1616   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
1617   }
1618
1619   /**
1620
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1666

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1665   } else {
1666   low = mid + 1;
1667   }
1668   }
1669
1670
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1670

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1669
1670   return high == 0 ? 0 : ckpts[high - 1].votes;
1671   }
1672
1673   /**
1674
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1782

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1781   uint256 pos = ckpts.length;
1782   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
1783   newWeight = op(oldWeight, delta);
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1785

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1786

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789   }
1790
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1793

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1792   function _add(uint256 a, uint256 b) private pure returns (uint256) {
1793   return a + b;
1794   }
1795
1796   function _subtract(uint256 a, uint256 b) private pure returns (uint256) {
1797
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1797

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1796   function _subtract(uint256 a, uint256 b) private pure returns (uint256) {
1797   return a - b;
1798   }
1799   }
1800
1801
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1851

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1850   if(block.timestamp >= unlockEnd) {
1851   return locked - claimed;
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1853

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
1856   /**
1857
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1853

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
1856   /**
1857
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1853

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
1856   /**
1857
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1853

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
1856   /**
1857
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1853

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1852   }
1853   return (locked * (block.timestamp - unlockBegin)) / (unlockEnd - unlockBegin) -
claimed;
1854   }
1855
1856   /**
1857
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1864

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1863   require(block.timestamp < unlockEnd, "TokenLock: Unlock period already complete");
1864   lockedAmounts[recipient] += amount;
1865   require(token.transferFrom(msg.sender, address(this), amount), "TokenLock:
Transfer failed");
1866   emit Locked(msg.sender, recipient, amount);
1867   }
1868
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1879

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1878   }
1879   claimedAmounts[msg.sender] += amount;
1880   require(token.transfer(recipient, amount), "TokenLock: Transfer failed");
1881   emit Claimed(msg.sender, recipient, amount);
1882   }
1883
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1913

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1912   uint256 constant teamSupply = 100_000_000e18;
1913   uint256 constant DAOTreasurySupply = 1_000_000_000e18 - airdropSupply -
teamSupply;
1914
1915   bool public vestStarted = false;
1916
1917
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1913

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1912   uint256 constant teamSupply = 100_000_000e18;
1913   uint256 constant DAOTreasurySupply = 1_000_000_000e18 - airdropSupply -
teamSupply;
1914
1915   bool public vestStarted = false;
1916
1917
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 388

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
387   unchecked {
388   counter._value = value - 1;
389   }
390   }
391
392
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1616

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1615   uint256 pos = _checkpoints[account].length;
1616   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
1617   }
1618
1619   /**
1620
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1670

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1669
1670   return high == 0 ? 0 : ckpts[high - 1].votes;
1671   }
1672
1673   /**
1674
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1782

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1781   uint256 pos = ckpts.length;
1782   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
1783   newWeight = op(oldWeight, delta);
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1785

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1786

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- L2DAOToken.sol

Locations

```
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789   }
1790
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
10
11   pragma solidity ^0.8.0;
12
13   /**
14   * @dev These functions deal with verification of Merkle Trees proofs.
15
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 66

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
65
66   pragma solidity ^0.8.0;
67
68   /**
69   * @dev Wrappers over Solidity's uintXX/intXX casting operators with added overflow
70
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 310

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
309
310   pragma solidity ^0.8.0;
311
312   /**
313   * @dev Standard math utilities missing in the Solidity language.
314
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 356

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
355
356   pragma solidity ^0.8.0;
357
358   /**
359   * @title Counters
360
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 402

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
401
402   pragma solidity ^0.8.0;
403
404   /**
405   * @dev String operations.
406
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 472

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
471
472   pragma solidity ^0.8.0;
473
474
475   /**
476
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 708

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
707
708   pragma solidity ^0.8.0;
709
710
711   /**
712
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 814

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
813
814   pragma solidity ^0.8.0;
815
816   /**
817    * @dev Interface of the ERC20 Permit extension allowing approvals to be made via
signatures, as defined in
818
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 877

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
876
877   pragma solidity ^0.8.0;
878
879   /**
880   * @dev Provides information about the current execution context, including the
881
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 904

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
903
904   pragma solidity ^0.8.0;
905
906
907   /**
908
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 982

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
981
982   pragma solidity ^0.8.0;
983
984   /**
985   * @dev Interface of the ERC20 standard as defined in the EIP.
986
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1067

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
1066
1067   pragma solidity ^0.8.0;
1068
1069
1070   /**
1071
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1097

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
1096
1097   pragma solidity ^0.8.0;
1098
1099
1100
1101
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1455

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
1454
1455   pragma solidity ^0.8.0;
1456
1457
1458
1459
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1544

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- L2DAOToken.sol

Locations

```
1543
1544   pragma solidity ^0.8.0;
1545
1546
1547
1548
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1804

low SEVERITY

The current pragma Solidity directive is ""^0.8.2"". 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

- L2DAOToken.sol

Locations

```
1803
1804   pragma solidity ^0.8.2;
1805
1806
1807   /**
1808
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1888

low SEVERITY

The current pragma Solidity directive is ""^0.8.2"". 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

- L2DAOToken.sol

Locations

```
1887
1888   pragma solidity ^0.8.2;
1889
1890
1891
1892
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 48

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
47   for (uint256 i = 0; i < proof.length; i++) {
48   bytes32 proofElement = proof[i];
49   if (computedHash <= proofElement) {
50   // Hash(current computed hash + current element of the proof)
51   computedHash = keccak256(abi.encodePacked(computedHash, proofElement));
52
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 429

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
428   digits -= 1;
429   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
430   value /= 10;
431   }
432   return string(buffer);
433
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 456

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
455   bytes memory buffer = new bytes(2 * length + 2);
456   buffer[0] = "0";
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 457

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
456   buffer[0] = "0";
457   buffer[1] = "x";
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 459

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461   }
462   require(value == 0, "Strings: hex length insufficient");
463
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 459

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
458   for (uint256 i = 2 * length + 1; i > 1; --i) {
459   buffer[i] = _HEX_SYMBOLS[value & 0xf];
460   value >>= 4;
461   }
462   require(value == 0, "Strings: hex length insufficient");
463
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1594

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1593   function checkpoints(address account, uint32 pos) public view virtual returns
(Checkpoint memory) {
1594   return _checkpoints[account][pos];
1595   }
1596
1597   /**
1598
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1616

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1615   uint256 pos = _checkpoints[account].length;
1616   return pos == 0 ? 0 : _checkpoints[account][pos - 1].votes;
1617   }
1618
1619   /**
1620
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1663

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1662   uint256 mid = Math.average(low, high);
1663   if (ckpts[mid].fromBlock > blockNumber) {
1664   high = mid;
1665   } else {
1666   low = mid + 1;
1667
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1670

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1669
1670   return high == 0 ? 0 : ckpts[high - 1].votes;
1671   }
1672
1673   /**
1674
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1782

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1781   uint256 pos = ckpts.length;
1782   oldWeight = pos == 0 ? 0 : ckpts[pos - 1].votes;
1783   newWeight = op(oldWeight, delta);
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1785

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1786

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- L2DAOToken.sol

Locations

```
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789   }
1790
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 1627

low SEVERITY

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

- L2DAOToken.sol

Locations

```
1626   function getPastVotes(address account, uint256 blockNumber) public view returns
(uint256) {
1627   require(blockNumber < block.number, "ERC20Votes: block not yet mined");
1628   return _checkpointsLookup(_checkpoints[account], blockNumber);
1629   }
1630
1631
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 1640

low SEVERITY

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

- L2DAOToken.sol

Locations

```
1639   function getPastTotalSupply(uint256 blockNumber) public view returns (uint256) {
1640   require(blockNumber < block.number, "ERC20Votes: block not yet mined");
1641   return _checkpointsLookup(_totalSupplyCheckpoints, blockNumber);
1642   }
1643
1644
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 1785

low SEVERITY

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

- L2DAOToken.sol

Locations

```
1784
1785   if (pos > 0 && ckpts[pos - 1].fromBlock == block.number) {
1786   ckpts[pos - 1].votes = SafeCast.toUint224(newWeight);
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 1788

low SEVERITY

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

- L2DAOToken.sol

Locations

```
1787   } else {
1788   ckpts.push(Checkpoint({fromBlock: SafeCast.toUint32(block.number), votes:
SafeCast.toUint224(newWeight)}));
1789   }
1790   }
1791
1792
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.