dYdX

Smart Contract Audit Report

SYSFIXED

13 Jun 2021

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
dYdX	DYDX	Binance Smart Chain

Addresses

Contract address	0x92d6c1e31e14520e676a687f0a93788b716beff5
Contract deployer address	0x301DF37d653b281AF83a1DDf4464eF21A622eC83

Project Website

https://dydx.exchange/

Codebase

https://etherscan.io/address/0x92d6c1e31e14520e676a687f0a93788b716beff5#code

SUMMARY

dYdX is a leading decentralized exchange that currently supports perpetual trading. dYdX runs on smart contracts on the Ethereum blockchain, and allows users to trade with no intermediaries.

Contract Summary

Documentation Quality

dYdX 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 dYdX 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 140, 174, 196, 197, 235, 273, 1018, 1038, 1085, 1086, 1095, 1097, 1097, 1097, 1104, 1154, 1157, 1160, 1365, 1387, 1691, 1729, 1018, 1038, 1085, 1086, 1095, 1104, 1154 and 1157.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 344.
- 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 1367, 1370, 1371, 1389, 1392 and 1393.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 925, 1074, 1147 and 1566.

CONCLUSION

We have audited the dYdX project released on June 2021 to discover issues and identify potential security vulnerabilities in dYdX 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 dYdX 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, the potential use of "block.number" as a source of randomness, a floating pragma is set, and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

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 Jun 12 2021 22:26:03 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Jun 13 2021 00:58:35 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	DydxToken.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	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-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-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 140

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
139   function add(uint256 a, uint256 b) internal pure returns (uint256) {
140   uint256 c = a + b;
141   require(c >= a, 'SafeMath: addition overflow');
142
143   return c;
144
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
173   require(b <= a, errorMessage);
174   uint256 c = a - b;
175
176   return c;
177   }
178
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 196

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
195
196   uint256 c = a * b;
197   require(c / a == b, 'SafeMath: multiplication overflow');
198
199   return c;
200
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 197

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
196   uint256 c = a * b;
197   require(c / a == b, 'SafeMath: multiplication overflow');
198
199   return c;
200   }
201
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 235

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
234   require(b > 0, errorMessage);
235   uint256 c = a / b;
236   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
237
238   return c;
239
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 273

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
272   require(b != 0, errorMessage);
273   return a % b;
274   }
275   }
276
277
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1018

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1017   if (fromSnapshotsCount != 0) {
1018   previous = snapshots[from][fromSnapshotsCount - 1].value;
1019   } else {
1020   previous = balanceOf(from);
1021   }
1022
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1038

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1037   if (toSnapshotsCount != 0) {
1038   previous = snapshots[to][toSnapshotsCount - 1].value;
1039   } else {
1040   previous = balanceOf(to);
1041   }
1042
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1085

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1084   // First check most recent balance
1085   if (snapshots[user][snapshotsCount - 1].blockNumber <= blockNumber) {
1086   return snapshots[user][snapshotsCount - 1].value;
1087   }
1088
1089
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1086

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1085   if (snapshots[user][snapshotsCount - 1].blockNumber <= blockNumber) {
1086   return snapshots[user][snapshotsCount - 1].value;
1087   }
1088
1089   // Next check implicit zero balance
1090
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1095

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1094   uint256 lower = 0;
1095   uint256 upper = snapshotsCount - 1;
1096   while (upper > lower) {
1097   uint256 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1098   Snapshot memory snapshot = snapshots[user][center];
1099
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1096   while (upper > lower) {
1097   uint256 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1098   Snapshot memory snapshot = snapshots[user][center];
1099   if (snapshot.blockNumber == blockNumber) {
1100   return snapshot.value;
1101
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1096   while (upper > lower) {
1097   uint256 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1098   Snapshot memory snapshot = snapshots[user][center];
1099   if (snapshot.blockNumber == blockNumber) {
1100   return snapshot.value;
1101
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1096   while (upper > lower) {
1097   uint256 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1098   Snapshot memory snapshot = snapshots[user][center];
1099   if (snapshot.blockNumber == blockNumber) {
1100   return snapshot.value;
1101
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1104

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1103   } else {
1104   upper = center - 1;
1105   }
1106   }
1107   return snapshots[user][lower].value;
1108
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1154

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1153   ownerSnapshotsCount != 0 &&
1154   ownerSnapshots[ownerSnapshotsCount - 1].blockNumber == currentBlock
1155   ) {
1156   // Doing multiple operations in the same block
1157   ownerSnapshots[ownerSnapshotsCount - 1].value = newValue;
1158
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1157

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1156   // Doing multiple operations in the same block
1157   ownerSnapshots[ownerSnapshotsCount - 1].value = newValue;
1158   } else {
1159   ownerSnapshots[ownerSnapshotsCount] = Snapshot(currentBlock, newValue);
1160   snapshotsCounts[owner] = ownerSnapshotsCount + 1;
1161
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1160

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1159   ownerSnapshots[ownerSnapshotsCount] = Snapshot(currentBlock, newValue);
1160   snapshotsCounts[owner] = ownerSnapshotsCount + 1;
1161   }
1162   }
1163
1164
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1365

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1364   {
1365   for (uint256 i = 0; i < addressesToAdd.length; i++) {
1366   require(
1367   !_tokenTransferAllowlist[addressesToAdd[i]],
1368   'ADDRESS_EXISTS_IN_TRANSFER_ALLOWLIST'
1369
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1387

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1386   {
1387   for (uint256 i = 0; i < addressesToRemove.length; i++) {
1388   require(
1389   _tokenTransferAllowlist[addressesToRemove[i]],
1390   'ADDRESS_DOES_NOT_EXIST_IN_TRANSFER_ALLOWLIST'
1391
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1691

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1690   require(
1691   nonce == _nonces[signer]++,
1692   'INVALID_NONCE'
1693   );
1694   require(
1695
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1729

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1728   require(
1729   nonce == _nonces[signer]++,
1730   'INVALID_NONCE'
1731   );
1732   require(
1733
```

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

LINE 1018

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1017   if (fromSnapshotsCount != 0) {
1018   previous = snapshots[from][fromSnapshotsCount - 1].value;
1019   } else {
1020   previous = balanceOf(from);
1021   }
1022
```

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

LINE 1038

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1037   if (toSnapshotsCount != 0) {
1038   previous = snapshots[to][toSnapshotsCount - 1].value;
1039   } else {
1040   previous = balanceOf(to);
1041   }
1042
```

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

LINE 1085

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1084   // First check most recent balance
1085   if (snapshots[user][snapshotsCount - 1].blockNumber <= blockNumber) {
1086   return snapshots[user][snapshotsCount - 1].value;
1087   }
1088
1089
```

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

LINE 1086

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1085   if (snapshots[user][snapshotsCount - 1].blockNumber <= blockNumber) {
1086   return snapshots[user][snapshotsCount - 1].value;
1087   }
1088
1089   // Next check implicit zero balance
1090
```

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

LINE 1095

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1094   uint256 lower = 0;
1095   uint256 upper = snapshotsCount - 1;
1096   while (upper > lower) {
1097   uint256 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1098   Snapshot memory snapshot = snapshots[user][center];
1099
```

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

LINE 1104

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1103   } else {
1104   upper = center - 1;
1105   }
1106   }
1107   return snapshots[user][lower].value;
1108
```

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

LINE 1154

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1153   ownerSnapshotsCount != 0 &&
1154   ownerSnapshots[ownerSnapshotsCount - 1].blockNumber == currentBlock
1155   ) {
1156   // Doing multiple operations in the same block
1157   ownerSnapshots[ownerSnapshotsCount - 1].value = newValue;
1158
```

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

LINE 1157

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1156   // Doing multiple operations in the same block
1157   ownerSnapshots[ownerSnapshotsCount - 1].value = newValue;
1158   } else {
1159   ownerSnapshots[ownerSnapshotsCount] = Snapshot(currentBlock, newValue);
1160   snapshotsCounts[owner] = ownerSnapshotsCount + 1;
1161
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 344

low SEVERITY

The current pragma Solidity directive is ""^0.7.5"". 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

- DydxToken.sol

Locations

```
343
344   pragma solidity ^0.7.5;
345
346
347
348
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1367

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1366   require(
1367   !_tokenTransferAllowlist[addressesToAdd[i]],
1368   'ADDRESS_EXISTS_IN_TRANSFER_ALLOWLIST'
1369   );
1370   _tokenTransferAllowlist[addressesToAdd[i]] = true;
1371
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1370

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1369   );
1370   _tokenTransferAllowlist[addressesToAdd[i]] = true;
1371   emit TransferAllowlistUpdated(addressesToAdd[i], true);
1372   }
1373   }
1374
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1371

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1370   _tokenTransferAllowlist[addressesToAdd[i]] = true;
1371   emit TransferAllowlistUpdated(addressesToAdd[i], true);
1372   }
1373   }
1374
1375
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1389

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1388   require(
1389   _tokenTransferAllowlist[addressesToRemove[i]],
1390   'ADDRESS_DOES_NOT_EXIST_IN_TRANSFER_ALLOWLIST'
1391   );
1392   _tokenTransferAllowlist[addressesToRemove[i]] = false;
1393
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1392

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1391   );
1392   _tokenTransferAllowlist[addressesToRemove[i]] = false;
1393   emit TransferAllowlistUpdated(addressesToRemove[i], false);
1394   }
1395   }
1396
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1393

low SEVERITY

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

Source File

- DydxToken.sol

Locations

```
1392   _tokenTransferAllowlist[addressesToRemove[i]] = false;
1393   emit TransferAllowlistUpdated(addressesToRemove[i], false);
1394   }
1395   }
1396
1397
```

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

LINE 925

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

- DydxToken.sol

Locations

```
924
925   return _searchByBlockNumber(snapshots, snapshotsCounts, user, block.number);
926   }
927
928   /**
929
```

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

LINE 1074

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

- DydxToken.sol

Locations

```
1073   require(
1074   blockNumber <= block.number,
1075   'INVALID_BLOCK_NUMBER'
1076   );
1077
1078
```

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

LINE 1147

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

- DydxToken.sol

Locations

```
1146   {
1147   uint128 currentBlock = uint128(block.number);
1148
1149   uint256 ownerSnapshotsCount = snapshotsCounts[owner];
1150   mapping(uint256 => Snapshot) storage ownerSnapshots = snapshots[owner];
1151
```

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

LINE 1566

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

- DydxToken.sol

Locations

```
1565   uint256 snapshotsCount = _totalSupplySnapshotsCount;
1566   uint128 currentBlock = uint128(block.number);
1567   uint128 newValue = uint128(totalSupply());
1568
1569   // Note: There is no special case for the total supply being updated multiple
times in the same
1570
```

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.