hi Dollar

Smart Contract Audit Report

SYSFIXED

22 Oct 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
hi Dollar	HI	Binance Smart Chain

Addresses

Contract address	0x77087ab5df23cfb52449a188e80e9096201c2097
Contract deployer address	0x7E700B8da2d3E1082F4F5a94F91532bdc6B89066

Project Website

https://hi.com/

Codebase

https://bscscan.com/address/0x77087ab5df23cfb52449a188e80e9096201c2097#code

SUMMARY

hi Reserve Limited (“Hi”) and its affiliates are creating the hi platform. All references to “Hi” in this document refer to the issuing entity and all references to “hi” in this document refer to the platform, unless otherwise expressly stated. Our aim is to bring an innovative range of mobile and online financial services to eligible users, leveraging blockchain technology. We intend to operate across the globe, with a strong focus on financial inclusion, security and compliance.

Contract Summary

Documentation Quality

hi Dollar 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 hi Dollar with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 735.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 397, 416, 438, 471, 473, 494, 495, 520, 522, 633, 725, 726, 726, 759, 762, 783, 783, 789, 789, 789, 793, 1176, 1176, 1178, 1186, 1194 and 726.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11, 37, 129, 213, 242, 599 and 642.
- 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 794.

CONCLUSION

We have audited the hi Dollar project released on October 2021 to discover issues and identify potential security vulnerabilities in the hi Dollar 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 hi Dollar 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, a state variable visibility is not 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. The current pragma Solidity directive is ""^0.8.0"". Specifying a fixed compiler version is recommended 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. For state variable visibility is not set, it is best practice to set the visibility of state variables explicitly. The default visibility for "approverCount" is internal. Other possible visibility settings are public and private.

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.	ISSUE FOUND
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.	PASS
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	Thursday Oct 21 2021 02:13:29 GMT+0000 (Coordinated Universal Time)
Finished	Friday Oct 22 2021 18:05:14 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	hiDollar.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	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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
396   unchecked {
397   _approve(sender, _msgSender(), currentAllowance - amount);
398   }
399
400   return true;
401
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
415   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
416   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
417   return true;
418   }
419
420
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 438

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
437   unchecked {
438   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
439   }
440
441   return true;
442
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 471

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
470   unchecked {
471   _balances[sender] = senderBalance - amount;
472   }
473   _balances[recipient] += amount;
474
475
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
472   }
473   _balances[recipient] += amount;
474
475   emit Transfer(sender, recipient, amount);
476
477
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 494

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
493
494   _totalSupply += amount;
495   _balances[account] += amount;
496   emit Transfer(address(0), account, amount);
497
498
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 495

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
494   _totalSupply += amount;
495   _balances[account] += amount;
496   emit Transfer(address(0), account, amount);
497
498   _afterTokenTransfer(address(0), account, amount);
499
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 520

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
519   unchecked {
520   _balances[account] = accountBalance - amount;
521   }
522   _totalSupply -= amount;
523
524
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
521   }
522   _totalSupply -= amount;
523
524   emit Transfer(account, address(0), amount);
525
526
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 633

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
632   unchecked {
633   _approve(account, _msgSender(), currentAllowance - amount);
634   }
635   _burn(account, amount);
636   }
637
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
724   uint256 public constant RATIO_DECIMALS = 4;  /** ratio decimals */
725   uint256 public constant RATIO_PRECISION = 10 ** RATIO_DECIMALS /** ratio precision?
10000 */;
726   uint256 public constant MAX_FEE_RATIO = 1 * RATIO_PRECISION - 1; /** max fee ratio,
100% */
727   uint256 public constant MIN_APPROVE_RATIO = 6666 ; /** min approve ratio, 66.66% */
728
729
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
725   uint256 public constant RATIO_PRECISION = 10 ** RATIO_DECIMALS /** ratio precision?
10000 */;
726   uint256 public constant MAX_FEE_RATIO = 1 * RATIO_PRECISION - 1; /** max fee ratio,
100% */
727   uint256 public constant MIN_APPROVE_RATIO = 6666 ; /** min approve ratio, 66.66% */
728
729   enum ApprovedStatus { NONE, STARTED, APPROVED, OPPOSED }
730
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 726

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
725   uint256 public constant RATIO_PRECISION = 10 ** RATIO_DECIMALS /** ratio precision?
10000 */;
726   uint256 public constant MAX_FEE_RATIO = 1 * RATIO_PRECISION - 1; /** max fee ratio,
100% */
727   uint256 public constant MIN_APPROVE_RATIO = 6666 ; /** min approve ratio, 66.66% */
728
729   enum ApprovedStatus { NONE, STARTED, APPROVED, OPPOSED }
730
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 759

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
758   Approvers[account] = true;
759   approverCount += 1;
760   } else {
761   delete Approvers[account];
762   approverCount -= 1;
763
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 762

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
761   delete Approvers[account];
762   approverCount -= 1;
763   }
764   emit ApproverChanged(account, approve);
765   }
766
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 783

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
782
783   return approvedCount * RATIO_PRECISION / approverCount >= MIN_APPROVE_RATIO;
784   }
785
786   function _isProposalOpposed(uint256 opposedCount) internal view returns(bool) {
787
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 783

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
782
783   return approvedCount * RATIO_PRECISION / approverCount >= MIN_APPROVE_RATIO;
784   }
785
786   function _isProposalOpposed(uint256 opposedCount) internal view returns(bool) {
787
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 789

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
788
789   return opposedCount  * RATIO_PRECISION / approverCount > RATIO_PRECISION -
MIN_APPROVE_RATIO;
790   }
791
792   function _existIn(address account, address[] memory accounts) internal pure
returns(bool) {
793
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 789

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
788
789   return opposedCount  * RATIO_PRECISION / approverCount > RATIO_PRECISION -
MIN_APPROVE_RATIO;
790   }
791
792   function _existIn(address account, address[] memory accounts) internal pure
returns(bool) {
793
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 789

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
788
789   return opposedCount  * RATIO_PRECISION / approverCount > RATIO_PRECISION -
MIN_APPROVE_RATIO;
790   }
791
792   function _existIn(address account, address[] memory accounts) internal pure
returns(bool) {
793
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 793

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
792   function _existIn(address account, address[] memory accounts) internal pure
returns(bool) {
793   for (uint256 i = 0; i < accounts.length; i++) {
794   if (account == accounts[i]) return true;
795   }
796   return false;
797
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1176

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
1175   require(amount > 0, "hiDollar: non-positive amount not allowed");
1176   uint256 fee = amount * _getUserFeeRatio(sender) / RATIO_PRECISION;
1177   if (fee > 0) {
1178   require(balanceOf(sender) >= amount + fee, "hiDollar: insufficient balance");
1179   }
1180
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1176

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
1175   require(amount > 0, "hiDollar: non-positive amount not allowed");
1176   uint256 fee = amount * _getUserFeeRatio(sender) / RATIO_PRECISION;
1177   if (fee > 0) {
1178   require(balanceOf(sender) >= amount + fee, "hiDollar: insufficient balance");
1179   }
1180
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1178

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
1177   if (fee > 0) {
1178   require(balanceOf(sender) >= amount + fee, "hiDollar: insufficient balance");
1179   }
1180
1181   super._transfer(sender, recipient, amount);
1182
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1186

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
1185   super._transfer(sender, address(this), fee);
1186   fees += fee;
1187   emit FrictionFee(sender, fee);
1188   }
1189   }
1190
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1194

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
1193   require(amount <= fees, "hiDollar: the amount exceeds the collectable fees");
1194   fees -= amount;
1195   super._transfer(address(this), msg.sender, amount);
1196   emit FeeCollected(feeCollector, amount);
1197   return true;
1198
```

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

LINE 726

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
725   uint256 public constant RATIO_PRECISION = 10 ** RATIO_DECIMALS /** ratio precision?
10000 */;
726   uint256 public constant MAX_FEE_RATIO = 1 * RATIO_PRECISION - 1; /** max fee ratio,
100% */
727   uint256 public constant MIN_APPROVE_RATIO = 6666 ; /** min approve ratio, 66.66% */
728
729   enum ApprovedStatus { NONE, STARTED, APPROVED, OPPOSED }
730
```

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

- hiDollar.sol

Locations

```
10
11   pragma solidity ^0.8.0;
12
13   /**
14   * @dev Provides information about the current execution context, including the
15
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 37

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

- hiDollar.sol

Locations

```
36
37   pragma solidity ^0.8.0;
38
39
40   /**
41
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 129

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

- hiDollar.sol

Locations

```
128
129   pragma solidity ^0.8.0;
130
131   /**
132   * @dev Interface of the ERC20 standard as defined in the EIP.
133
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 213

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

- hiDollar.sol

Locations

```
212
213   pragma solidity ^0.8.0;
214
215
216   /**
217
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 242

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

- hiDollar.sol

Locations

```
241
242   pragma solidity ^0.8.0;
243
244
245
246
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 599

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

- hiDollar.sol

Locations

```
598
599   pragma solidity ^0.8.0;
600
601
602
603
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 642

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

- hiDollar.sol

Locations

```
641
642   pragma solidity ^0.8.2;
643
644   /**
645   * @see
https://docs.soliditylang.org/en/latest/contracts.html?highlight=experimental#return-
variables
646
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 735

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "approverCount" is internal. Other possible visibility settings are public and private.

Source File

- hiDollar.sol

Locations

```
734   mapping (address => bool) public Approvers;
735   uint256 approverCount;
736
737   mapping (address => bool) public Proposers;
738
739
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 794

low SEVERITY

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

Source File

- hiDollar.sol

Locations

```
793   for (uint256 i = 0; i < accounts.length; i++) {
794   if (account == accounts[i]) return true;
795   }
796   return false;
797   }
798
```

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.