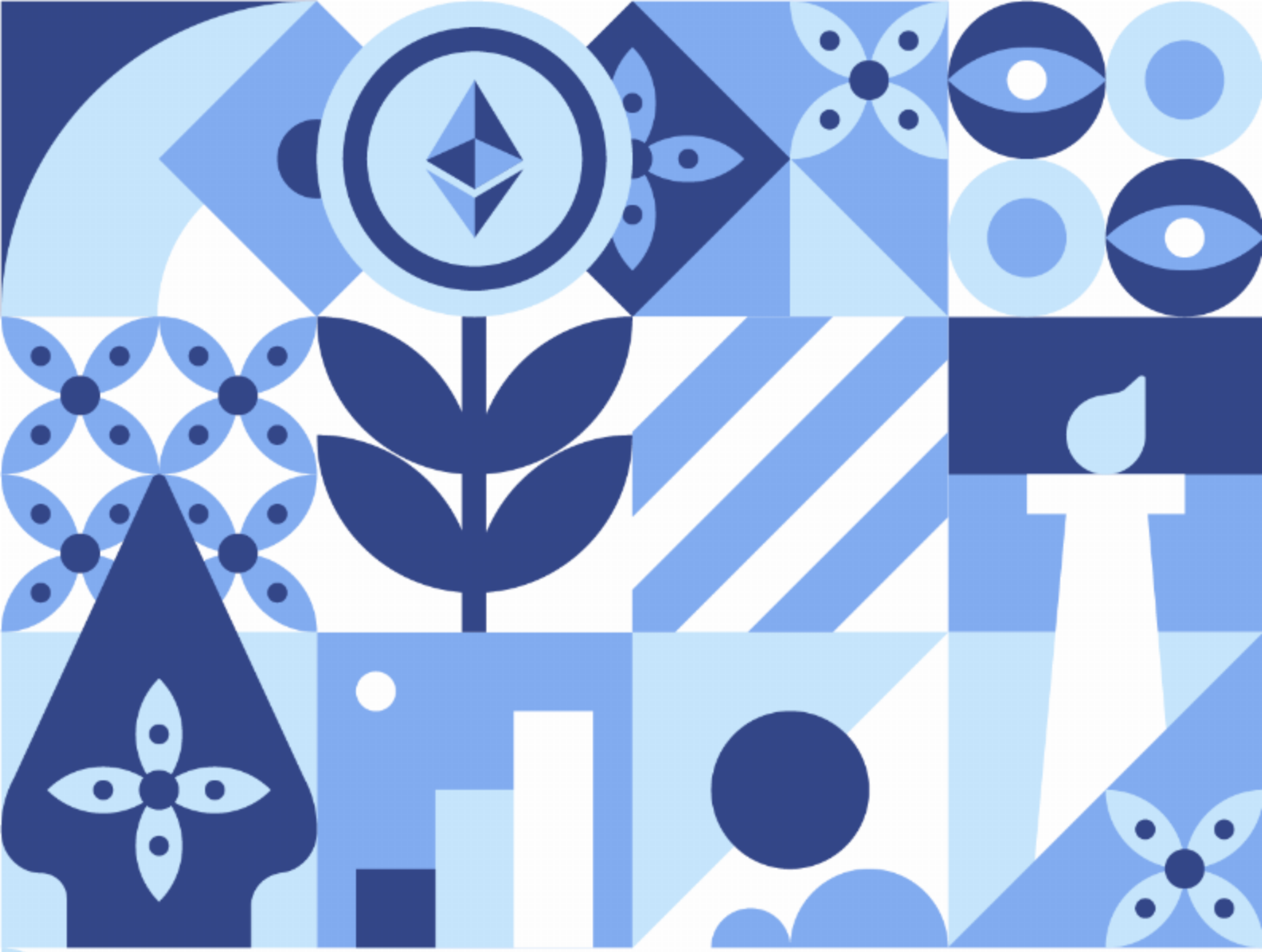

Shyft Network

# Smart Contract Audit Report

SYSFIXED

30 Jun 2021

# TABLE OF CONTENTS

# AUDITED DETAILS

## Audited Project

| Project name | Token ticker | Blockchain |
|---|---|---|
| Shyft Network | SHFT | Ethereum |

## Addresses

| | |
|---|---|
| **Contract address** | 0xb17C88bDA07D28B3838E0c1dE6a30eAfBCF52D85 |
| **Contract deployer address** | 0x208Af4ee6FBE1f767a362633A70162237fa3Aac6 |

## Project Website

**https://www.shyft.network/**

## Codebase

**https://etherscan.io/address/0xb17C88bDA07D28B3838E0c1dE6a30eAfBCF52D85#code**

# SUMMARY

Shyft Network is a public protocol designed to validate identity and power compliance directly into blockchain data. By facilitating the transfer of verifiable data between centralized and decentralized ecosystems, Shyft Network delivers meaningful user information that institutions can utilize to secure cryptocurrency while maintaining privacy.

## Contract Summary

**Documentation Quality**

Shyft Network 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 Shyft Network 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 1376, 1378, 1380, 1383, 1385, 1387, 1390 and 1413.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 310, 311, 319, 425, 426, 436, 986, 998, 1011, 1012, 1023, 1033, 1047, 1064, 1079, 1080, 1098, 1115, 1133, 1153, 1173, 2153, 2153, 2153, 425 and 426.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 5.
- 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 431, 434, 476, 2153, 2153, 2153 and 2153.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1711, 1713 and 1720.

# CONCLUSION

We have audited the Shyft Network project released on June 2021 to discover issues and identify potential security vulnerabilities in Shyft Network 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 a satisfactory result with some low-risk issues.

The issues found in the Shyft Network 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, tx.origin as a part of authorization control 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<br>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<br>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<br>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. | ISSUE FOUND |
| Block values as a proxy for time | SWC-116 | Block numbers should not be used for time calculations. | PASS |
| Signature Unique ID | SWC-117<br>SWC-121<br>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<br>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** | Tuesday Jun 29 2021 02:15:22 GMT+0000 (Coordinated Universal Time) |
| **Finished** | Wednesday Jun 30 2021 17:37:55 GMT+0000 (Coordinated Universal Time) |
| **Mode** | Standard |
| **Main Source File** | ShyftKycContract.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 | 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-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |

| | | | |
|---|---|---|---|
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET. | low | acknowledged |
| SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL. | low | acknowledged |
| SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL. | low | acknowledged |
| SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL. | 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-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 310

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
309
310   uint256 constant TestNetTokenOffset = 2**128;
311   uint256 constant PrivateNetTokenOffset = 2**192;
312
313   uint256 constant ShyftTokenType = 7341;
314
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 311

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
310   uint256 constant TestNetTokenOffset = 2**128;
311   uint256 constant PrivateNetTokenOffset = 2**192;
312
313   uint256 constant ShyftTokenType = 7341;
314   uint256 constant EtherTokenType = 60;
315
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 319

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
318   //Shyft Testnets
319   uint256 constant BridgeTownTokenType = TestNetTokenOffset + 0;
320
321   //Ethereum Testnets
322   uint256 constant GoerliTokenType = 5;
323
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 425

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
424
425   uint256 toDeleteIndex = valueIndex - 1;
426   uint256 lastIndex = set._values.length - 1;
427
428   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
429
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 426

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
425   uint256 toDeleteIndex = valueIndex - 1;
426   uint256 lastIndex = set._values.length - 1;
427
428   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
429   // so rarely, we still do the swap anyway to avoid the gas cost of adding an 'if'
statement.
430
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 436

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
435   // Update the index for the moved value
436   set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are 1-based
437
438   // Delete the slot where the moved value was stored
439   set._values.pop();
440
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 986

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
985   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
986   uint256 c = a + b;
987   if (c < a) return (false, 0);
988   return (true, c);
989   }
990
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 998

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
997   if (b > a) return (false, 0);
998   return (true, a - b);
999   }
1000
1001   /**
1002
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1011

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1010   if (a == 0) return (true, 0);
1011   uint256 c = a * b;
1012   if (c / a != b) return (false, 0);
1013   return (true, c);
1014   }
1015
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1012

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1011   uint256 c = a * b;
1012   if (c / a != b) return (false, 0);
1013   return (true, c);
1014   }
1015
1016
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1023

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1022   if (b == 0) return (false, 0);
1023   return (true, a / b);
1024   }
1025
1026   /**
1027
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1033

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1032   if (b == 0) return (false, 0);
1033   return (true, a % b);
1034   }
1035
1036   /**
1037
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1047

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1046   function add(uint256 a, uint256 b) internal pure returns (uint256) {
1047   uint256 c = a + b;
1048   require(c >= a, "SafeMath: addition overflow");
1049   return c;
1050   }
1051
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1064

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1063   require(b <= a, "SafeMath: subtraction overflow");
1064   return a - b;
1065   }
1066
1067   /**
1068
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1079

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1078   if (a == 0) return 0;
1079   uint256 c = a * b;
1080   require(c / a == b, "SafeMath: multiplication overflow");
1081   return c;
1082   }
1083
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1080

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1079   uint256 c = a * b;
1080   require(c / a == b, "SafeMath: multiplication overflow");
1081   return c;
1082   }
1083
1084
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1098

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1097   require(b > 0, "SafeMath: division by zero");
1098   return a / b;
1099   }
1100
1101   /**
1102
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1115

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1114   require(b > 0, "SafeMath: modulo by zero");
1115   return a % b;
1116   }
1117
1118   /**
1119
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1133

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1132   require(b <= a, errorMessage);
1133   return a - b;
1134   }
1135
1136   /**
1137
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1152   require(b > 0, errorMessage);
1153   return a / b;
1154   }
1155
1156   /**
1157
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1173

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1172   require(b > 0, errorMessage);
1173   return a % b;
1174   }
1175   }
1176
1177
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

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

LINE 425

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
424
425   uint256 toDeleteIndex = valueIndex - 1;
426   uint256 lastIndex = set._values.length - 1;
427
428   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
429
```

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

LINE 426

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
425   uint256 toDeleteIndex = valueIndex - 1;
426   uint256 lastIndex = set._values.length - 1;
427
428   // When the value to delete is the last one, the swap operation is unnecessary.
However, since this occurs
429   // so rarely, we still do the swap anyway to avoid the gas cost of adding an 'if'
statement.
430
```

# SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

## low SEVERITY

The current pragma Solidity directive is ""^0.7.1"". 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

- ShyftKycContract.sol

## Locations

```
4
5   pragma solidity ^0.7.1;
6   //SPDX-License-Identifier: UNLICENSED
7
8   /* New ERC23 contract interface */
9
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1376

## low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "totalSupplyBip32X" is internal. Other possible visibility settings are public and private.

## Source File

- ShyftKycContract.sol

## Locations

```
1375   /// @dev Mapping of total supply specific bip32x assets.
1376   mapping(uint256 => uint256) totalSupplyBip32X;
1377   /// @dev Mapping of users to their balances of specific bip32x assets.
1378   mapping(address => mapping(uint256 => uint256)) balances;
1379   /// @dev Mapping of users to users with amount of allowance set for specific
bip32x assets.
1380
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1378

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1377   /// @dev Mapping of users to their balances of specific bip32x assets.
1378   mapping(address => mapping(uint256 => uint256)) balances;
1379   /// @dev Mapping of users to users with amount of allowance set for specific
bip32x assets.
1380   mapping(address => mapping(address => mapping(uint256 => uint256))) allowed;
1381
1382
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1380

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1379   /// @dev Mapping of users to users with amount of allowance set for specific
bip32x assets.
1380   mapping(address => mapping(address => mapping(uint256 => uint256))) allowed;
1381
1382   /// @dev Mapping of users to whether they have set auto-upgrade enabled.
1383   mapping(address => bool) autoUpgradeEnabled;
1384
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1383

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1382   /// @dev Mapping of users to whether they have set auto-upgrade enabled.
1383   mapping(address => bool) autoUpgradeEnabled;
1384   /// @dev Mapping of users to whether they Accepts Kyc Input only.
1385   mapping(address => bool) onlyAcceptsKycInput;
1386   /// @dev Mapping of users to whether their Accepts Kyc Input option is locked
permanently.
1387
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1385

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1384   /// @dev Mapping of users to whether they Accepts Kyc Input only.
1385   mapping(address => bool) onlyAcceptsKycInput;
1386   /// @dev Mapping of users to whether their Accepts Kyc Input option is locked
permanently.
1387   mapping(address => bool) lockOnlyAcceptsKycInputPermanently;
1388
1389
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1387

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1386   /// @dev Mapping of users to whether their Accepts Kyc Input option is locked
permanently.
1387   mapping(address => bool) lockOnlyAcceptsKycInputPermanently;
1388
1389   /// @dev mutex lock, prevent recursion in functions that use external function
calls
1390   bool locked;
1391
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1390

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1389   /// @dev mutex lock, prevent recursion in functions that use external function
calls
1390   bool locked;
1391
1392   /// @dev Whether there has been an upgrade from this contract.
1393   bool public hasBeenUpdated;
1394
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1413

## low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "nativeBip32X_type" is internal. Other possible visibility settings are public and private.

## Source File

- ShyftKycContract.sol

## Locations

```
1412   /// @dev The native Bip32X type of this network. Ethereum is 60, Shyft is 7341,
etc.
1413   uint256 nativeBip32X_type;
1414
1415   /// @dev The name of the minter role for implementing AccessControl
1416   bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");
1417
```

# SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1711

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1710   // burn tokens in this contract
1711   uint256 existingOriginBalance = balances[tx.origin][nativeBip32X_type];
1712
1713   balances[tx.origin][nativeBip32X_type] = 0;
1714   totalSupplyBip32X[nativeBip32X_type] =
totalSupplyBip32X[nativeBip32X_type].sub(existingOriginBalance);
1715
```

# SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1713

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1712
1713   balances[tx.origin][nativeBip32X_type] = 0;
1714   totalSupplyBip32X[nativeBip32X_type] =
totalSupplyBip32X[nativeBip32X_type].sub(existingOriginBalance);
1715
1716   //~70k gas for the contract "call"
1717
```

# SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1720

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
1719
1720   bool didTransferOrigin = migrateToKycContract(updatedShyftKycContractAddress,
tx.origin, existingOriginBalance.add(msg.value));
1721
1722   if (didTransferOrigin == true) {
1723
1724
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 431

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
430
431   bytes32 lastvalue = set._values[lastIndex];
432
433   // Move the last value to the index where the value to delete is
434   set._values[toDeleteIndex] = lastvalue;
435
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 434

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
433   // Move the last value to the index where the value to delete is
434   set._values[toDeleteIndex] = lastvalue;
435   // Update the index for the moved value
436   set._indexes[lastvalue] = toDeleteIndex + 1; // All indexes are 1-based
437
438
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 476

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
475   require(set._values.length > index, "EnumerableSet: index out of bounds");
476   return set._values[index];
477   }
478
479   // Bytes32Set
480
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2153

## low SEVERITY

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

## Source File

- ShyftKycContract.sol

## Locations

```
2152   if (_data.length >= 4) {
2153   tokenSig = bytes4(uint32(bytes4(bytes1(_data[3])) >> 24) +
uint32(bytes4(bytes1(_data[2])) >> 16) + uint32(bytes4(bytes1(_data[1])) >> 8) +
uint32(bytes4(bytes1(_data[0]))));
2154   }
2155
2156   // reject the transaction if the token signature is a "withdrawToExternalContract"
event from the v0 contract.
2157
```

# 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.