VITAL BLOCK

Security Assessment

Intswap

Vital Block Verified on August 10th, 2023

@Vital-Block

@VB_Audit

info@vitalblock.org

www.vitalblock.org

PREPARED FOR:

INTSWAP

INTRODUCTION

Auditing Firm	**VITAL BLOCK SECURITY**
Client Firm	**INTSWAP FINANCE**
Methodology	Automated Analysis, Manual Code Review
Language	SOL
Contract Code	IntswapV1CreatorManager.sol IntswapV1Factory.sol IntswapV1Pair.sol IntswapV1Permision.sol IntswapV1RoyaltyVault.sol IntswapV1StakingCenter.sol IntswapV1TimelockController.sol incentives BuyIncentiveStrategy.sol FixedRewardRateStrategy.sol RoyaltyDistributionStrategy.sol
Blockchain	zkSync Era
Centralization	Active ownership
Website	https://intswap.io/
Discord	https://discord.gg/cvPJAz2Bms
Twitter	https://twitter.com/Intswap_amm
GitHub	https://docs.intswap.io/
Prelim Report Date	August 9th , 2023
Final Report Date	August 10th 2023

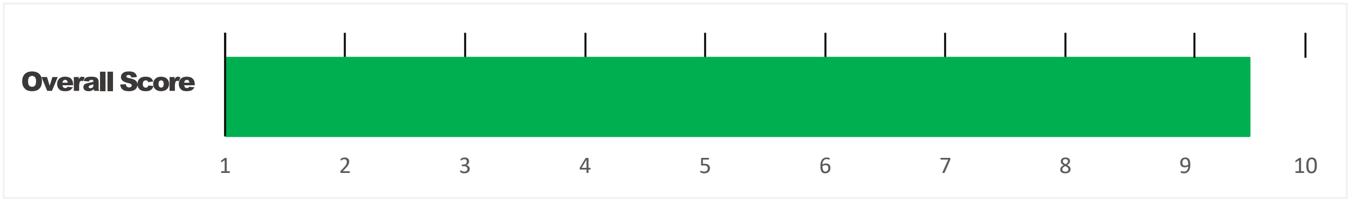
i Verify the authenticity of this report on our GitHub Repo: https://www.github.com/vital-block

EXECUTIVE SUMMARY

INTSWAP has performed the automated and manual analysis of the INTSWAP Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical ! ●	Major " ●	Medium # ●	Minor $ ●	Unknown % ●
Open	0	0	1	2	0
Acknowledged	0	0	2	3	0
Informational	0	0	0	3	0
Noteworty onlyOwner Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

INTSWAP Smart contract has achieved the following score: 95.0



Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

SCOPE OF WORK

Vital Block Security was consulted by INTSWAP to conduct the smart contract audit of its. Sol source code. The audit scope of work is strictly limited to mentioned .SOL file only..

External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

Contracts checked	
IntswapV1CreatorManager.sol **IntswapV1Factory.sol** **IntswapV1Pair.sol** **IntswapV1Permision.sol** **IntswapV1RoyaltyVault.sol** **IntswapV1StakingCenter.sol** **IntswapV1TimelockController.sol** **incentives** **BuyIncentiveStrategy.sol** **FixedRewardRateStrategy.sol** **RoyaltyDistributionStrategy.sol**	
Project Name	**INTSWAP FINANCE**
Token Symbol	**INT**

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block auditing process and methodology:

CONNECT

- **The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.**

AUDIT

- **Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:**
  - **Remix IDE Developer Tool**
  - **Open Zeppelin Code Analyzer**
  - **SWC Vulnerabilities Registry**
  - **DEX Dependencies, e.g., Pancakeswap, Uniswap**
- **Simulations are performed to identify centralized exploits causing contract and/or trade locks.**
- **A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:**

Centralized Exploits	○ **Token Supply Manipulation** ○ **Access Control and Authorization** ○ **Assets Manipulation** ○ **Ownership Control** ○ **Liquidity Access** ○ **Stop and Pause Trading** ○ **Ownable Library Verification**

ℹ️ Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to the codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

i It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- **Privileged roles can be granted the power to `pause()` the contract in case of an external attack.**
- **Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUDIT SCOPE | INTSWAP

ID	Repo	Comment	File	SHM211 Checksum
ITM	intswap-core-v1-zksync /contracts	cC51D65	IntswapV1CreatorManager.sol	67515802c6be0fd50f8632d8433cccc9d b6f4b39f9e566d1fa78de54b84bddr54
IRY	intswap-core-v1-zksync /contracts	cC51D53	IntswapV1Factory.sol	890ppkjjjk96be0fd50f8632d8433cccc9 db6f4b39f9e566d1yhhg8765fffckiuybb
ITV	intswap-core-v1-zksync /contracts	cC51D61	IntswapV1Pair.sol	12KI6778uj908766362fvyga98jdkl8864 8yhfbqt37409owehbgwhuyyyg223738
IML	intswap-core-v1-zksync /contracts	cC51D76	IntswapV1Pair.sol	98uuyriy399787390uhbiiuhghhdg7guu 30oi7799u9359ydfgdgygeigi3ioueyy78
ITR	intswap-core-v1-zksync /contracts	cC51D22	IntswapV1Permision.sol	0566efgywqutfeuh87872t1537883798 3639293763hhegetgjfwjk89336668862
IOP	intswap-core-v1-zksync /contracts	cC51D44	IntswapV1Permision.sol	766363ttebnve88329973mvvdsggct47 8153ytgdfdxy792635fgdjgi1900990908
IDP	intswap-core-v1-zksync /contracts	cC51D21	IntswapV1RoyaltyVault.sol	835656990327hudbinnjntr6729dchjld0 993ytyy3vq63235727879889073
IWY	intswap-core-v1-zksync /contracts	cC51D97	IntswapV1RoyaltyVault.sol	cc089692343d1cc36eaf196046d7a528 d153abd55ba20e82f1d57c22fcd92675
IKB	intswap-core-v1-zksync /contracts	cC51D76	IntswapV1StakingCenter.sol	8448b3af42497f5f74e53424ee3e6c55 1f51356945108d22a893d608a7990542
IXY	intswap-core-v1-zksync /contracts	cC51D23	IntswapV1StakingCenter.sol	5c86aa1dd3889db5fcd17a80214b226f c784f268ab9db82df97c1d2459467831
ICB	intswap-core-v1-zksync /contracts	cC51D63	IntswapV1TimelockController.sol	b8244da33db171e5533d77bef4a3570 3df1de2cebea5f35cb38ce6a26c778cf1
IWO	intswap-core-v1-zksync /contracts	cC51D60	IntswapV1TimelockController.sol	3d408b8f2cc56f9699a402b5151de906 71de089c3007afc9e4fc867c04152e7c
IGT	intswap-core-v1-zksync/contracts /incentives/	cC51D54	BuyIncentiveStrategy.sol	9d751621c3501102e4b50005ca3314ec 6e04e6ff8bbb30852d1c7edfff3f8cef
IDF	intswap-core-v1-zksync/contracts /incentives/	cC51D78	BuyIncentiveStrategy.sol	455687gfesadjknlppiuhhg774580vgfxr ki9876dhgvb990lkjhde444566788
IHC	intswap-core-v1-zksync/contracts /incentives/	cC51D80	FixedRewardRateStrategy.sol	78fhjkkkjeuuuibndjdnmkowete8a7889 wujdjokmskjuwhdddeeroi098hdua
ILP	intswap-core-v1-zksync/contracts /incentives/	cC51D56	FixedRewardRateStrategy.sol	6839yhdtwoimcb7263fvxsmlkoiaqwpo ye7gbcgefdd632sdetg21097hbr
IGB	intswap-core-v1-zksync/contracts /incentives/	cC51D50	RoyaltyDistributionStrategy.sol	234098uionakldfrb3576hgfdvei6ghdvb e8921yuowefdjjklpouowetg54376
INT	intswap-core-v1-zksync/contracts /incentives/	cC51D92	RoyaltyDistributionStrategy.sol	782043dtwjblopnbvdreswfvuklopresa qcxzsfgtryiingdmiretdkpotrdy6790

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💵	Function is payable
🔒	Function is internal
🔐	Function is private
❗	Function is important

```
| **INTSWAP** | Interface |   |||
| └ | totalSupply | External ▯ |        ❗     |NO▯ |
| └ | decimals | External ▯ |        ❗     |NO▯ |
| └ | symbol | External ▯ |        ❗     |NO▯ |
| └ | name | External ▯ |        ❗     |NO▯ |
| └ | getOwner | External ▯ |              |NO▯ |
| └ | balanceOf | External ▯ |     ❗          |NO▯ |
| └ | transfer | External ▯ | "     ❗     🛑     |NO▯ |
| └ | allowance | External ▯ |     ❗          |NO▯ |
| └ | approve | External ▯ | "     ❗     🛑     |NO▯ |
| └ | transferFrom | External ▯ | "              |NO▯ |
||||||
| **IFactoryV2** | Interface |        |||
| └ | getPair | External ▯ |              |NO▯ |
| └ | createPair | External ▯ | "              |NO▯ |
||||||
| **IV2Pair** | Interface |        |||
| └ | factory | External ▯ |              |NO▯ |
| └ | getReserves | External ▯ |              |NO▯ |
| └ | sync | External ▯ | "              |NO▯ |
||||||
| **IRouter01** | Interface |  |||
| └ | factory | External  |  |NO |
| └ | SOL | External  |  |NO |
| └ | addLiquiditySol | External  |  # |NO |
| └ | addLiquidity | External  | " |NO |
| └ | swapExactSolForTokens | External  |  # |NO |
| └ | getAmountsOut | External  |  |NO |
| └ | getAmountsIn | External  |  |NO |
||||||
| **IRouter02** | Interface | IRouter01 |||
| └ | swapExactTokensForSolSupportingFeeOnTransferTokens | External  | " |NO |
| └ | swapExactSolForTokensSupportingFeeOnTransferTokens | External  |  # |NO |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External  | " ❗ 🛑 |NO |
| └ | swapExactTokensForTokens | External  | " |NO |
||||||
| **Protections** | Interface |  |||
| └ | checkUser | External  | " ❗ 🛑 |NO |
| └ | setLaunch | External  | " |NO |
| └ | setLpPair | External  | " |NO |
| └| INT | External  | " |NO |
| └ | removeSniper | External  | " |NO |
||||||
| **Cashier** | Interface |  |||
| └ | setRewardsProperties | External  | " |NO |
| └ | tally | External  | " |NO |
| └ | load | External  |  # |NO |
| └ | cashout | External  | " |NO |
| └ | giveMeWelfarePlease | External  | " |NO |
| └ | getTotalDistributed | External  |  |NO |
| └ | getUserInfo | External  |  |NO |
| └ | getUserRealizedRewards | External  |  |NO |
```

| └ | getPendingRewards | External ▯ | |NO▯ |

| └ | initialize | External ▯ | " |NO▯ |

| └ | getCurrentReward | External ▯ | |NO▯ |

||||||

| **SOL** | Implementation | **SafeMath** |||

| └ | <Constructor> | Public ▯ | # |NO▯ |

| └ | transferOwner | External ▯ | " | onlyOwner |

| └ | renounceOwnership | External ▯ | " | NO❗|

| └ | setOperator | Public ▯ | " |NO▯ |

| └ | renounceOriginalDeployer | External ▯ | " |NO▯ |

| └ | <Receive Sui> | External ▯ | # |NO▯ |

| └ | totalSupply | External ▯ | |NO▯ |

| └ | decimals | External ▯ | |NO▯ |

| └ | symbol | External ▯ | |NO▯ |

| └ | name | External ▯ | |NO▯ |

| └ | getOwner | External ▯ | ❗ |NO▯ |

| └ | balanceOf | Public ▯ | ❗ |NO▯ |

| └ | allowance | External ▯ | ❗ |NO▯ |

| └ | approve | External ▯ | " ❗ 🛑 |NO▯ |

| └ | _approve | Internal $ | " 🔐 🛑 | |

| └ | approveContractContingency | Public ▯ | " ❗ 🛑 | onlyOwner |

| └ | transfer | External ▯ | " |NO▯ |

| └ | transferFrom | External ▯ | " |NO▯ |

| └ | setNewRouter | External ▯ | " | onlyOwner |

| └ | setLpPair | External ▯ | " | onlyOwner |

| └ | setInitializers | External ▯ | " | onlyOwner |

| └ | isExcludedFromFees | External ▯ | |NO▯ |

| └ | isExcludedFromDividends | External ▯ | |NO▯ |

| └ | isExcludedFromProtection | External ▯ | |NO▯ |

| └ | setDividendExcluded | Public ▯ | " | onlyOwner |

| └ | setExcludedFromFees | Public ▯ | " | onlyOwner |

OWV-01 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	intswap-core-v1-zksync/contracts /IntswapV1CreatorManager.sol	Acknowledged

Description

In **updateForAddress,** the following equation is used inside an unchecked block

```
  constructor(address _royaltyVault, IIntswapV1TimelockController _timelockController)
{
        royaltyVault = _royaltyVault;
        timelockController = _timelockController;
    }

    function updateFactory(IIntswapV1Factory _factory) external onlyOwner {
        address oldFactory = address(factory);
        factory = _factory;
```

Where parameters. Address Out Used is a this and override In is a this.
As these two are multiplied together in an unchecked block, they may overflow.

Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never causean overflow

OZT-02 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	Iintswap-core-v1-zksync/contracts /ntswapV1Pair.sol	Acknowledged

Description

In **UpdateForToken,** the following equation is used inside an unchecked block

```
  function pruneOtherNFTs(IERC721 _token, uint16[] memory _tokenIds) external onlyFactory
{
        require(_token != nft, "IntswapV1Pair: Not allow to prune reserve token");
        for (uint256 i; i < _tokenIds.length; i++) {
            _token.transferFrom(address(this), msg.sender, _tokenIds[i]);
        }
```

Owner can not issue more **INT** tokens indefinitely.
Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the INT contract.

Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never cause an overflow.

OHT-03 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Inconsistency	**Informational**	intswap-core-v1-zksync/contracts /IntswapV1Permision.sol	Informational

Description

In **UpdateForMapping,** the following equation is used inside an unchecked block

```
struct Permision {
    bool isOnlyEOA;
    bool isOnlyOwner;
    bool isWhitelist;
    mapping(address => mapping(bytes32 => bool)) whitelist;
    mapping(address => mapping(bytes32 => bool)) blacklist;
}

mapping(address => Permision) public permisions;
```

The function **Mapping** () does not have the override specifier. It should be noted that since **Mapping** > a function that overrides only a single interface function does not require the override specifier (see doc). However, all other instances of this in the codebase contain the override specifier

Recommendation

We recommend either checking for overflow in this case, or ensuring that the PairsIn is close enough it will never cause an overflow.

STV-04 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Status Mathematical Operations	**Minor**	intswap-core-v1-zksync/contracts /IntswapV1RoyaltyVault.sol	INFORMATIONAL

Description

State variables can be declared as constant using the constant keyword. This means that the **Reward** of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
  function updateRewardStrategy(IRoyaltyDistributionStrategy _rewardStrategy) external
onlyOwner {
        address oldRewardStrategy = address(rewardStrategy);
        rewardStrategy = _rewardStrategy;

        emit NewRewardStrategy(oldRewardStrategy, address(_rewardStrategy))
```

Recommendation

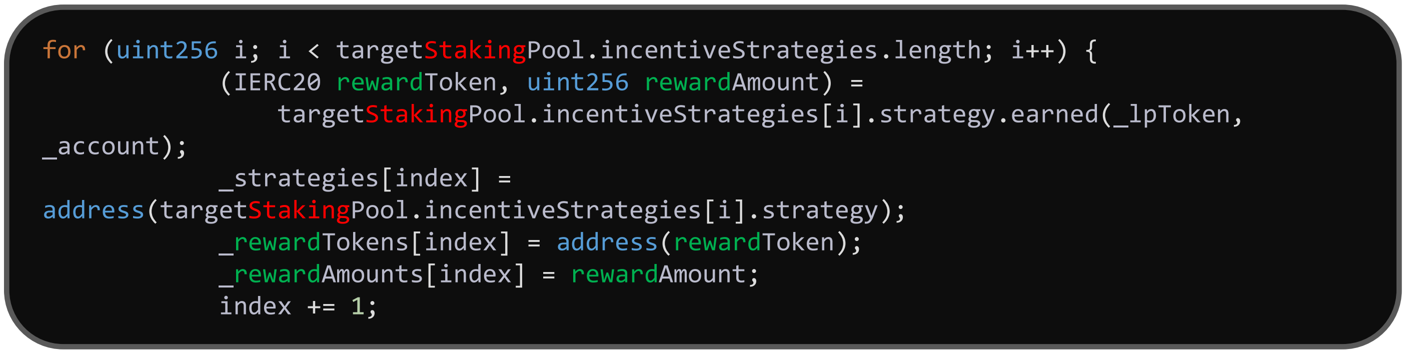
Constant state variables can be useful when the contract wants to ensure that the **Reward** of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

GZT-05 POSSIBLE OVERFLOW

Category	Severity ●	Location	Status
Inconsistency	**Informational**	intswap-core-v1-zksync/contracts /IntswapV1StakingCenter.sol	Acknowledged

Description

In **UpdateForStaking ,** the following equation is used inside an unchecked block



```
for (uint256 i; i < targetStakingPool.incentiveStrategies.length; i++) {
            (IERC20 rewardToken, uint256 rewardAmount) =
                targetStakingPool.incentiveStrategies[i].strategy.earned(_lpToken,
_account);
            _strategies[index] =
address(targetStakingPool.incentiveStrategies[i].strategy);
            _rewardTokens[index] = address(rewardToken);
            _rewardAmounts[index] = rewardAmount;
            index += 1;
```

Recommendation

We recommend either checking for overflow in this case, or ensuring that the Staking is close enough it will never cause an overflow.

OPTIMIZATIONS | INTSWAP

ID	Title	Category	Status
OTV	**Logarithm Refinement Optimization**	Gas Optimization	Acknowledged ●
OKP	**Checks Can Be Performed Earlier**	Gas Optimization	Acknowledged ●
ODP	**Unnecessary Use Of SafeMath**	Gas Optimization	Acknowledged ●
OWY	**Struct Optimization**	Gas Optimization	Acknowledged ●
OGT	**Unused State Variable**	Gas Optimization	Acknowledged ●

General Detectors

Missing Zero Address Validation

Some functions in this contract may not appropriately check for zero addresses being used.

Attention Required

Numeric Notation Best Practices

The numeric notation used in this contract is unconventional, possibly worsening the reading/debugging experience.

Attention Required

- ✓ No compiler version inconsistencies found
- ✓ No unchecked call responses found
- ✓ No vulnerable self-destruct functions found
- ✓ No assertion vulnerabilities found
- ✓ No old solidity code found
- ✓ No external delegated calls found
- ✓ No external call dependency found
- ✓ No vulnerable authentication calls found
- ✓ No invalid character typos found
- ✓ No RTL characters found
- ✓ No dead code found
- ✓ No risky data allocation found
- ✓ No uninitialized state variables found
- ✓ No uninitialized storage variables found
- ✓ No vulnerable initialization functions found
- ✓ No risky data handling found
- ✓ No number accuracy bug found
- ✓ No out-of-range number vulnerability found
- ✓ No map data deletion vulnerabilities found
- ✓ No tautologies or contradictions found
- ✓ No faulty true/false values found
- ✓ No innacurate divisions found
- ✓ No redundant constructor calls found
- ✓ No vulnerable transfers found
- ✓ No vulnerable return values found
- ✓ No uninitialized local variables found
- ✓ No default function responses found
- ✓ No missing arithmetic events found
- ✓ No missing access control events found
- ✓ No redundant true/false comparisons found
- ✓ No state variables vulnerable through function calls found
- ✓ No buggy low-level calls found
- ✓ No expensive loops found
- ✓ No bad numeric notation practices found
- ✓ No missing constant declarations found
- ✓ No missing external function declarations found
- ✓ No vulnerable payable functions found
- ✓ No vulnerable message values found

Vulnerability Scan

REENTRANCY

✓ No reentrancy risk found

Severity	Minor
Confidence Parameter	Certain

Vulnerability Description

Not Mintable: A large amount of this token can not be minted by a private wallet or contract.

Scanning Line:

```
function hook(bytes32 action, bytes memory data) external
{
        require(msg.sender == factory,
"BuyIncentiveStrategy: Only factory.");

        if (action == keccak256("Buy")) {
            (address trader, address lpToken, ,uint256
baseTokenInputAmount, , ) =
                abi.decode(data, (address, address,
uint256, uint256, uint256, uint256));

            _updateRewards(lpToken, trader,
baseTokenInputAmount);
        }
    }

    function getRewards() external returns (uint256
totalRewards) {
        for (uint256 i; i < officalPairs.length; i++) {
            totalRewards += getReward(officalPairs[i]);
        }
    }
```

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor

```
contract FixedRewardRateStrategy is IIntswapV1IncentiveStrategy,
AccessControl, ReentrancyGuard {
    bytes32 public constant ADMIN_ROLE = keccak256("ADMIN_ROLE");
    bytes32 public constant CREATE_NEW_INCENTIVE =
keccak256("CREATE_NEW_INCENTIVE");
    string public name = "FixedRewardRateStrategy"
```

Description:

Floating point calculations can vary across different architectures.

Alleviation:

This exhibit was acknowledged and ultimately discarded by the **INTSWAP** team due to low severity. We consider the exhibit fully attended to as it doesn't impose any meaningful security concerns.

RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.

Repository:

https://github.com/Intswap-Labs/intswap-core-v1-zksync/tree/main/contracts

All Audited Files:

IntswapV1CreatorManager.sol

IntswapV1Factory.sol

IntswapV1Pair.sol

IntswapV1Permision.sol

IntswapV1RoyaltyVault.sol

IntswapV1StakingCenter.sol

IntswapV1TimelockController.sol

incentives

BuyIncentiveStrategy.sol

FixedRewardRateStrategy.sol

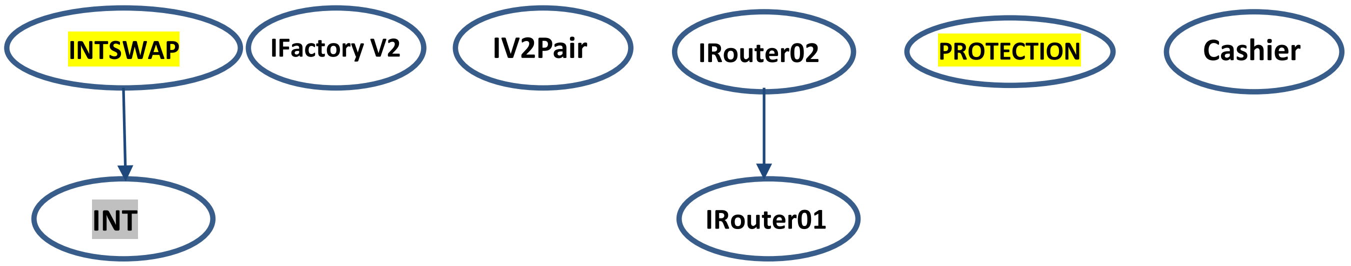
RoyaltyDistributionStrategy.sol

Contracts:

Contract File:

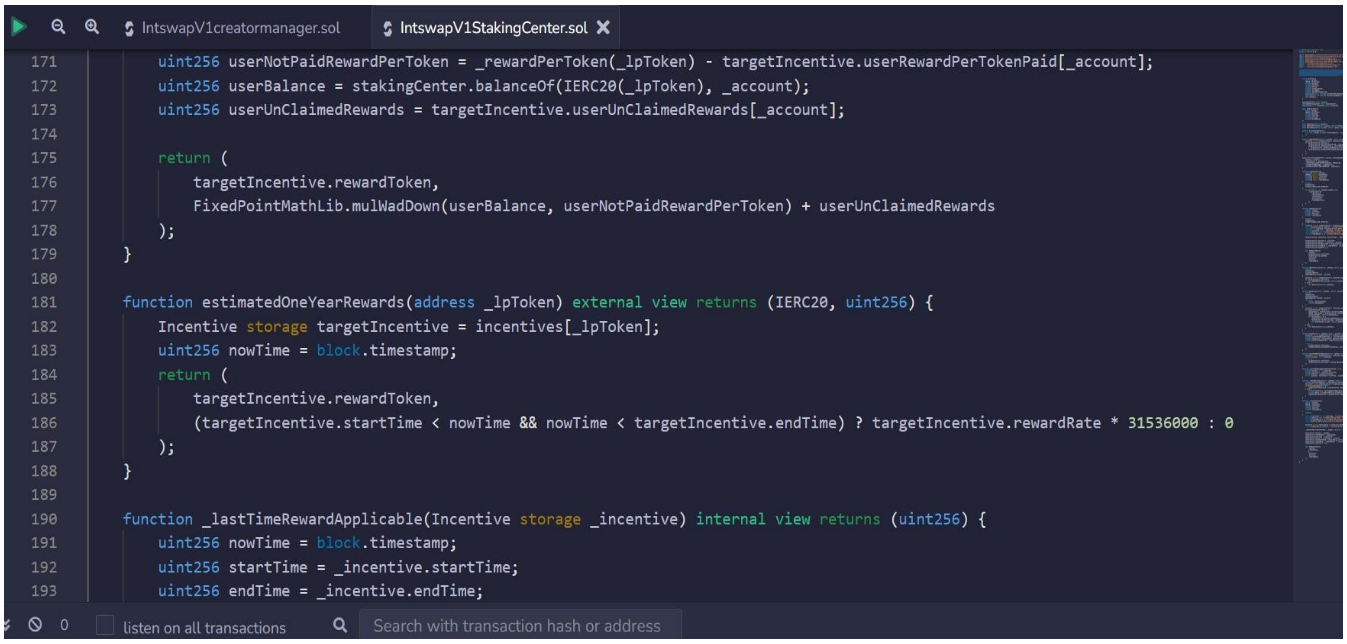
```
intswap-core-v1-zksync/contracts
/IntswapV1CreatorManager.sol

intswap-core-v1-zksync/contracts
/IntswapV1Factory.sol

intswap-core-v1-zksync/contracts
/IntswapV1Pair.sol

intswap-core-v1-zksync/contracts
/IntswapV1Permision.sol

intswap-core-v1-zksync/contracts
/IntswapV1RoyaltyVault.sol

intswap-core-v1-zksync/contracts
/IntswapV1StakingCenter.sol

intswap-core-v1-zksync/contracts
/IntswapV1TimelockController.sol

intswap-core-v1-zksync/contracts/incentives
/BuyIncentiveStrategy.sol

intswap-core-v1-zksync/contracts/incentives
/FixedRewardRateStrategy.sol

intswap-core-v1-zksync/contracts/incentives
/RoyaltyDistributionStrategy.sol
```

INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of INTSWAP	Medium # 🟡

Vulnerability 0 : No important security issue detected.
Threat level: Low


IntswapV1creatormanager.sol
IntswapV1StakingCenter.sol
listen on all transactions
Search with transaction hash or address

```
        uint256 userNotPaidRewardPerToken = _rewardPerToken(_lpToken) - targetIncentive.userRewardPerTokenPaid[_account];
        uint256 userBalance = stakingCenter.balanceOf(IERC20(_lpToken), _account);
        uint256 userUnClaimedRewards = targetIncentive.userUnClaimedRewards[_account];

        return (
            targetIncentive.rewardToken,
            FixedPointMathLib.mulWadDown(userBalance, userNotPaidRewardPerToken) + userUnClaimedRewards
        );
    }

    function estimatedOneYearRewards(address _lpToken) external view returns (IERC20, uint256) {
        Incentive storage targetIncentive = incentives[_lpToken];
        uint256 nowTime = block.timestamp;
        return (
            targetIncentive.rewardToken,
            (targetIncentive.startTime < nowTime && nowTime < targetIncentive.endTime) ? targetIncentive.rewardRate * 31536000 : 0
        );
    }

    function _lastTimeRewardApplicable(Incentive storage _incentive) internal view returns (uint256) {
        uint256 nowTime = block.timestamp;
        uint256 startTime = _incentive.startTime;
        uint256 endTime = _incentive.endTime;
```

MANUAL REVIEW

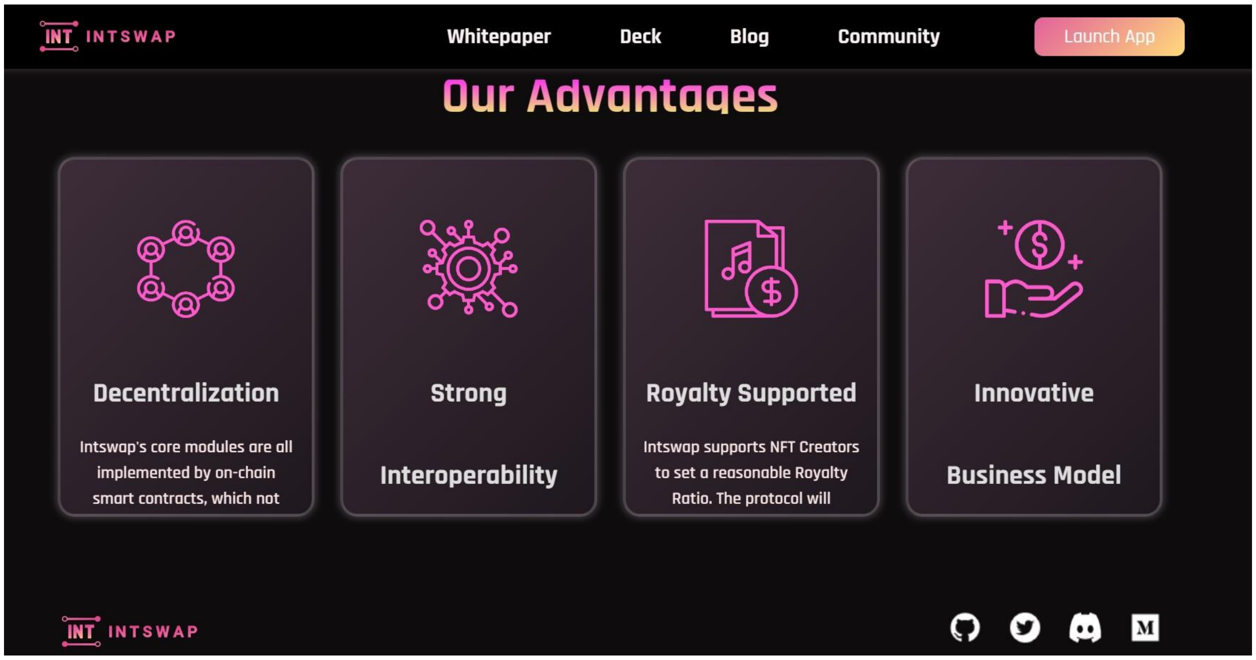
INTSWAP: is the 1st zkSync Era NFT AMM Protocol enables LP Mining to earn compound trading fee, royalty fee and beyond.
Intswap is a decentralized NFT AMM. Users can easily exchange between Fungible Token and Non Fungible Token from Intswap. In addition, Intswap has greatly reduced the user's transaction slippage cost through the original AMM with dynamic concentrated liquidity function, making it more suitable for the transaction characteristics of NFT Marketplace.

TOKEN NAME: INTSWAP
Ticker: INT

Chain/Standard: Zksync Network

LAUNGUGE: SOL

The INTSWAP Platform Is Launching ZKSYNC Blockchain


INT INTSWAP
Whitepaper
Deck
Blog
Community
Launch App
Our Advantages
Decentralization
Intswap's core modules are all implemented by on-chain smart contracts, which not
Strong Interoperability
Royalty Supported
Intswap supports NFT Creators to set a reasonable Royalty Ratio. The protocol will
Innovative Business Model
INT INTSWAP

VitalBlock

ISSUES CHECKING STATUS

	Issue Description	Checking Status
1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	SOL Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the Move Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon Move contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED

AUDIT RESULT

PASSED

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor ●

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

```
        }
  function updateRewardStrategy(IRoyaltyDistributionStrategy _rewardStrategy) external
onlyOwner {
        address oldRewardStrategy = address(rewardStrategy);
        rewardStrategy = _rewardStrategy;
```

RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.

RECOMMENDATION

Deployer and/or contract owner private keys are secured carefully.

Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

ALLEVIATION

The INTSWAP project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project

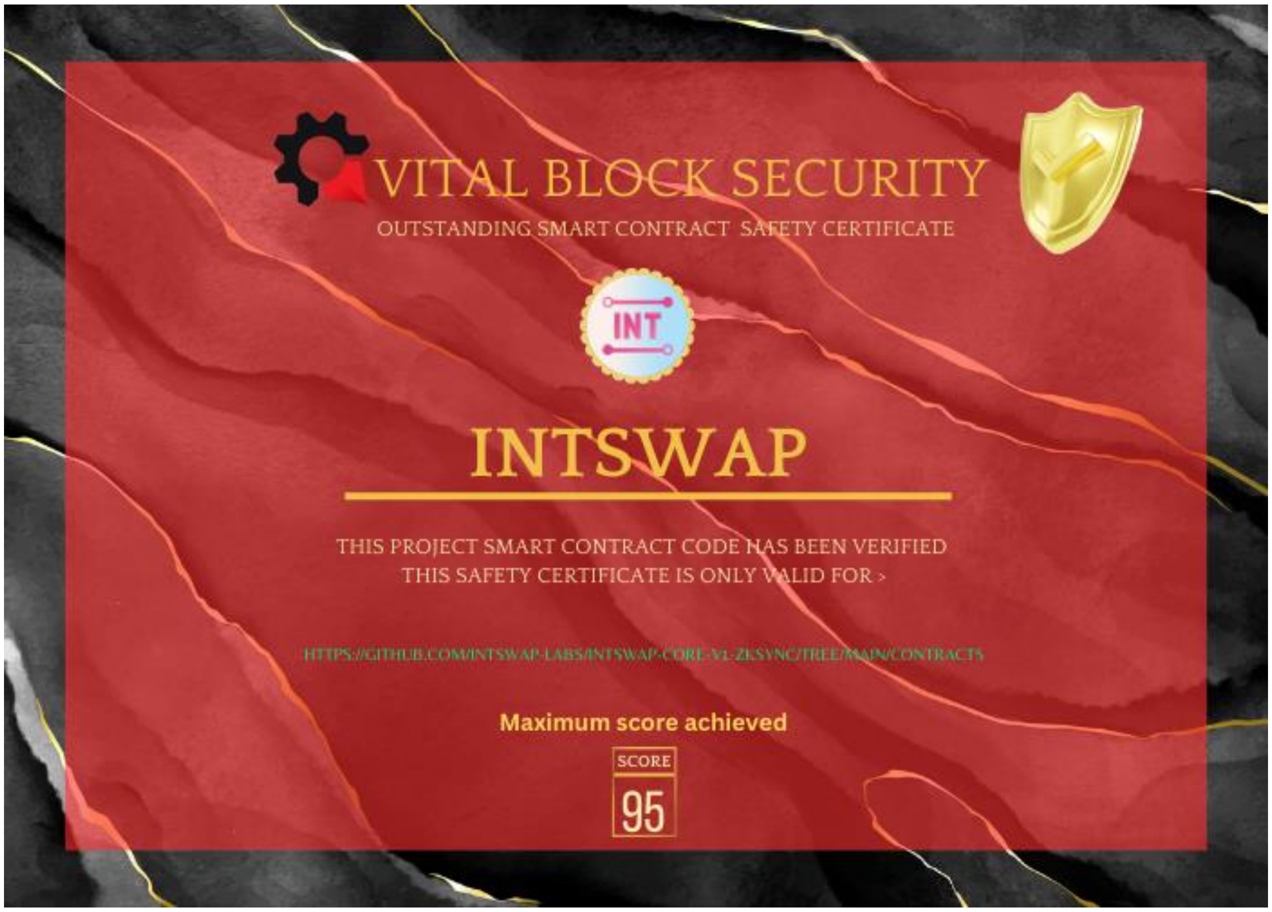
Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor ●

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.

CERTIFICATE BY VITAL BLOCK SECURITY


VITAL BLOCK SECURITY

OUTSTANDING SMART CONTRACT SAFETY CERTIFICATE

INT

INTSWAP

THIS PROJECT SMART CONTRACT CODE HAS BEEN VERIFIED
THIS SAFETY CERTIFICATE IS ONLY VALID FOR >

HTTPS://GITHUB.COM/INTSWAP-LABS/INTSWAP-CORE-V1-ZKSYNC/TREE/MAIN/CONTRACTS

Maximum score achieved

SCORE

95

DISCLAIMERS

Vital Block provides the easy-to-understand audit of Solidity, Move and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHERWEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT VITAL BLOCK

Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.

Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.

Website: https://Vitalblock.org

Email: info@vitalblock.org

GitHub: https://github.com/vital-block

Telegram (Engineering): https://t.me/vital_block

Telegram (Onboarding): https://t.me/vitalblock_cmo

vital-block

info@vitalblock.org

www.Vitalblock.org

VITALBLOCK SMART CONTRACT AUDIT

Vital Block Dedicated to securing Public and Private Blockchain Ecosystem