VITAL BLOCK

Security Assessment

STERLING

Vital Block Verified on Feb 27th, 2023

@Vital-Block

@VB_Audit

info@vitalblock.org

www.vitalblock.org

PREPARED FOR:

STERLING

INTRODUCTION

Auditing Firm	**VITAL BLOCK SECURITY**
Client Firm	**STERLING FINANCE**
Methodology	**Automated Analysis, Manual Code Review**
Language	**Solidity**
Contract	**TOKEN:** 0x5DB7b150c5F38c5F5db11dCBDB885028fcC51D68 **factory:** 0xF7A23B9A9dCB8d0aff67012565C5844C20C11AFC **router:** 0x0cBD3aEa90538a1Cf3C60B05582b691f6d2b2B01 **controller:** 0x4d01714A025b4308F01b0f0E1FEb560673f863Ef **gaugesFactory**: 0xe4032AB13c189ac58f745DEb57B0b23868B50DCd **bribesFactory**: 0x03ed94D06f6E290b372cB7D72A5f885718abEf6f **ve**: 0x450330Df68E1ed6e0683373D684064bDa9115fEe **voter**: 0x474E967717B7A12352e1Cb731492bcc01d7816e2 **minter**: 0xF329eD282354DD6ea88e7149C1A169EB467908c6
Blockchain	**ARBITRUM**
Centralization	**Active ownership**
Website	**https://www.sterling.finance/**
Discord	**http://discord.gg/x5XVzBU8Ub**
Twitter	**https://twitter.com/Sterling_Fi**
GitHub	**https://github.com/Sterl-o/sterligfinancecontracts**
Prelim Report Date	**February 24, 2023**
Final Report Date	**February 25, 2023**

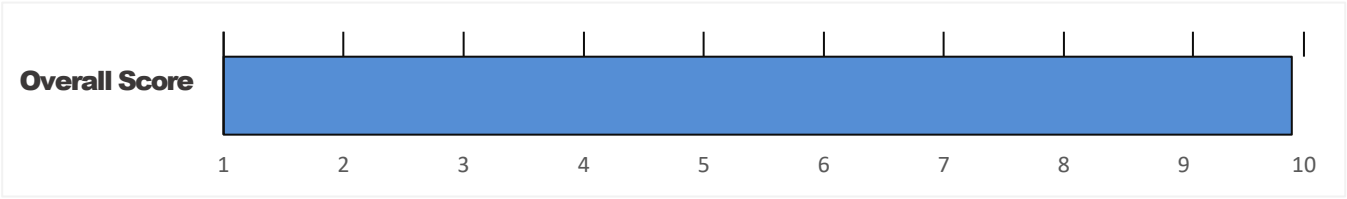
i Verify the authenticity of this report on our GitHub Repo: https://www.github.com/vital-block

EXECUTIVE SUMMARY

STERLING has performed the automated and manual analysis of the 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	0	3	0
Acknowledged	0	0	2	4	0
Resolved	0	0	0	0	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				

STERLING Smart contract has achieved the following score: 98.8



ℹ **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.**

ℹ **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 was consulted by STERLING to conduct the smart contract audit of its. Sol source code. The audit scope of work is strictly limited to mentioned .SOL file only:

- **STERLING.Sol**

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:

Public Contract Link	
0x5DB7b150c5F38c5F5db11dCBDB885028fcC51D68	
Contract Name	**STERLING**
Token Symbol	**STR**
Decimals	**18**
Total Supply	**1,005,000**

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**

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 | STERLING.FINANCE

ID	Repo	Comment	File	SHM211 Checksum
STM	contracts/base/core	cC51D68	GovernanceTreasury.sol	85f15802c6be0fd50f8632d8433cccc9db6f4b39f9e566d1fa78de54b84bdd35
SRY	contracts/base/core	cC51D68	PairFees.sol	8oippkjjjk96be0fd50f8632d8433cccc9db6f4b39f9e566d1yhhg8765fffckiuybb
STV	contracts/base/core	cC51D68	StrFactory.sol	3666778uj908766362fvyga98jdkl88648yhfbqt37409owehbgwhuyyyg223738
SML	contracts/base/core	cC51D68	StrPair.sol	98uuyriy399787390uhbiiuhghhdg7guu30oi7799u9359ydfgdgygeigi3ioueyy78
STR	contracts/base/token	cC51D68	Str.sol	4566efgywqutfeuh87872t15378837983639293763hhegetgjfwjk89336668862
SOP	contracts/base/token	cC51D68	StrMinter.sol	546363ttebnve88329973mvvdsggct478153ytgdfdxy792635fgdjgi1900990908
SDP	contracts/base/vote	cC51D68	StrVoter.sol	835656990327hudbinnjntr6729dchjld0993ytyy3vq63235727879889073
SWY	contracts/base/vote	cC51D68	Ve.sol	cc089692343d1cc36eaf196046d7a528d153abd55ba20e82f1d57c22fcd92675
SKB	contracts/base/vote	cC51D68	VeDist.sol	8448b3af42497f5f74e53424ee3e6c551f51356945108d22a893d608a7990542
SXY	contracts/base/reward	cC51D68	Bribe.sol	5c86aa1dd3889db5fcd17a80214b226fc784f268ab9db82df97c1d2459467831
SCB	contracts/base/reward	cC51D68	BribeFactory.sol	b8244da33db171e5533d77bef4a35703df1de2cebea5f35cb38ce6a26c778cf1
SWO	contracts/base/reward	cC51D68	Gauge.sol	3d408b8f2cc56f9699a402b5151de90671de089c3007afc9e4fc867c04152e7c
SGT	contracts/base/reward	cC51D68	GaugeFactory.sol	9d751621c3501102e4b50005ca3314ec6e04e6ff8bbb30852d1c7edfff3f8cef
SFF	contracts/base/reward	cC51D68	MultiRewardsPoolBase.sol	455687gfesadjknlppiuhhg774580vgfxrki9876dhgvb990lkjhde444566788

AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	**Function is internal**
	**Function is private**
	Function is important

STERLING	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❗
└	ETH	External ❗	|NO❗
└	addLiquidityETH	External ❗	# |NO❗
└	addLiquidity	External ❗	" |NO❗
└	swapExactAPTForTokens	External ❗	# |NO❗
└	getAmountsOut	External ❗	|NO❗
└	getAmountsIn	External ❗	|NO❗
IRouter02	Interface	IRouter01		
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	" |NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	# |NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	" ! 🛑 |NO❗
└	swapExactTokensForTokens	External ❗	" |NO❗
Protections	Interface			
└	checkUser	External ❗	" ! 🛑 |NO❗
└	setLaunch	External ❗	" |NO❗
└	setLpPair	External ❗	" |NO❗
└	STR	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 Ether> | 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 |

STV-03 POSSIBLE OVERFLOW

Category	Severity	Location	Status
StatusMathematical Operations	**Minor**	contracts/base/core/StrFactory.sol	Acknowledged

Description

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

```
mapping(address => mapping(address => mapping(bool => address))) public override getPair;
address[] public allPairs;
```

Where parameters.amountOutUsed is a mapping and override In is a mapping. 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

SRY-02 MISSING OVERRIDE SPECIFIER

Category	Severity	Location	Status
Inconsistency	**Informational**	contracts/base/core/PairFees.sol	Acknowledged

```
function claimFeesFor(address recipient, uint amount0, uint amount1) external {
require(msg.sender == pair, "Not pair");
if (amount0 > 0) {
IERC20(token0).safeTransfer(recipient, amount0)
```

Description

The function amount0 () does not have the override specifier. It should be noted that since amount0 > a function thatoverrides only a single interface function does not require the override specifier (see doc). However, all other instances of thisin the codebase contain the override specifier

Recommendation

We recommend adding the override specifier to amount() or removing the override specifier from all other functions thisapplies to for consistancy

OPTIMIZATIONS | STERLING

ID	Title	Category	Status
STV	**Logarithm Refinement Optimization**	Gas Optimization	Acknowledged
SOP	**Checks Can Be Performed Earlier**	Gas Optimization	Acknowledged
SDP	**Unnecessary Use Of SafeMath**	Gas Optimization	Acknowledged
SWY	**Struct Optimization**	Gas Optimization	Acknowledged
SGT	**Unused State Variable**	Gas Optimization	Acknowledged

Vulnerability Scan

REENTRANCY

Severity	Major
Confidence Parameter	Certain

Vulnerability Description

NOTE: In a re-entrance attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Scanning Line:

```
import "../Reentrancy.sol";

// The base pair of pools, either stable or volatile
contract StrPair is IERC20, IPair, Reentrancy {
 using SafeERC20 for IERC20;

 string public name;
 string public symbol;
 uint8 public constant decimals = 18;

 /// @dev Used to denote stable or volatile pair
 bool public immutable stable;

 uint public override totalSupply = 0;

 mapping(address => mapping(address => uint)) public override allowance;
 mapping(address => uint) public override balanceOf;

 bytes32 public immutable DOMAIN_SEPARATOR;
 // keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)");
 bytes32 public constant PERMIT_TYPEHASH =
0x6e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c64845d6126c9;
 uint internal constant _FEE_PRECISION = 1e32;
 mapping(address => uint) public nonces;
 uint public immutable chainId;
```

Repository:

https://github.com/Sterl-o/sterligfinancecontracts

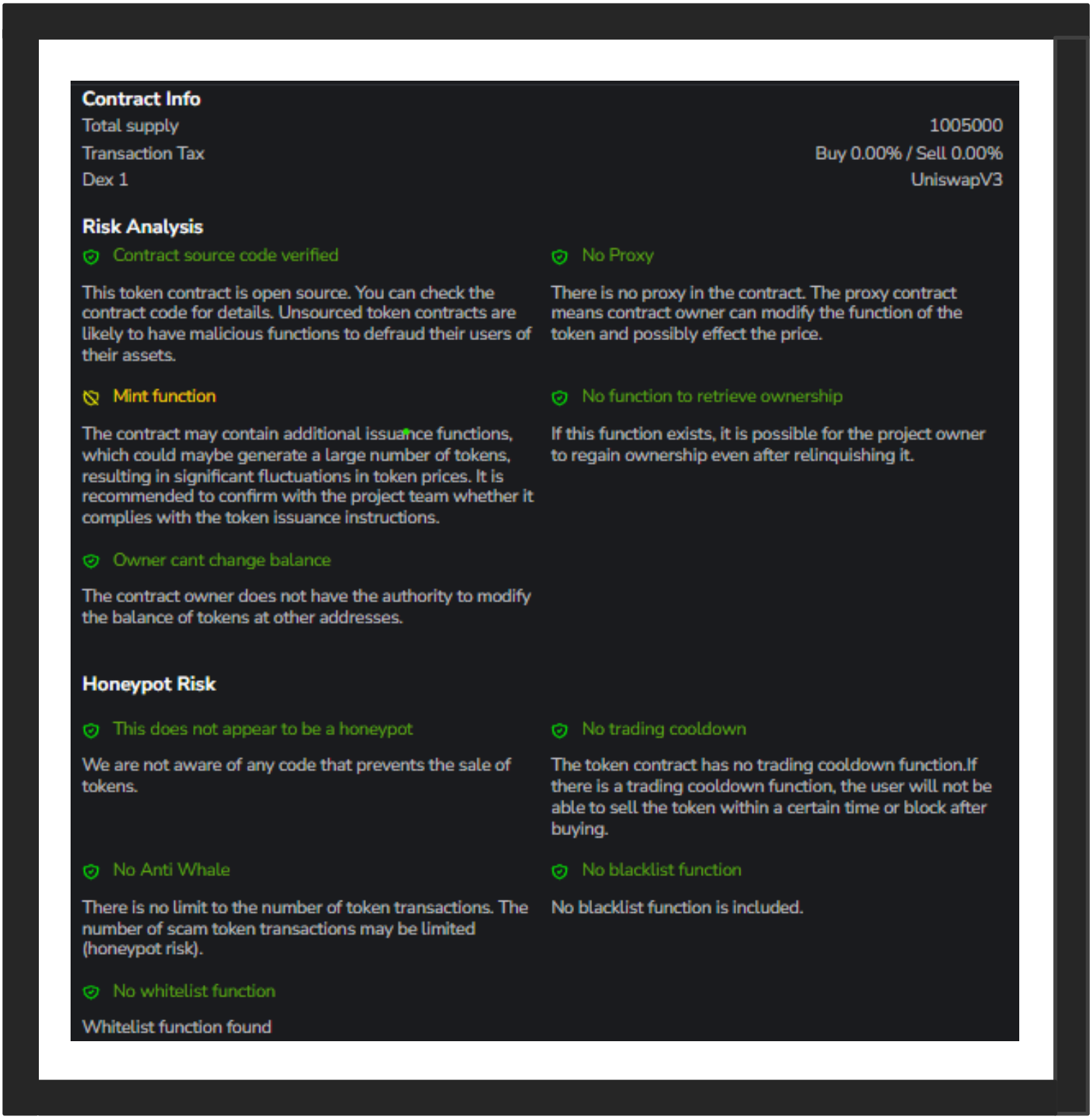
All Audited Files

Factory.sol
Router.sol
Controller.sol
Token.sol
gaugesFactory.sol
bribesFactory.sol
Ve.sol
Voter.sol
Minter.sol

Contract

Contract:
factory: 0xF7A23B9A9dCB8d0aff67012565C5844C20C11AFC
router: 0x0cBD3aEa90538a1Cf3C60B05582b691f6d2b2B01
controller: 0x4d01714A025b4308F01b0f0E1FEb560673f863Ef
token: 0x5DB7b150c5F38c5F5db11dCBDB885028fcC51D68
gaugesFactory: 0xe4032AB13c189ac58f745DEb57B0b23868B50DCd
bribesFactory: 0x03ed94D06f6E290b372cB7D72A5f885718abEf6f
ve: 0x450330Df68E1ed6e0683373D684064bDa9115fEe
voter: 0x474E967717B7A12352e1Cb731492bcc01d7816e2
minter: 0xF329eD282354DD6ea88e7149C1A169EB467908c6

Vulnerability Run check



Contract Info

Total supply	1005000
Transaction Tax	Buy 0.00% / Sell 0.00%
Dex 1	UniswapV3

Risk Analysis

Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

No Proxy

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

Mint function

The contract may contain additional issuance functions, which could maybe generate a large number of tokens, resulting in significant fluctuations in token prices. It is recommended to confirm with the project team whether it complies with the token issuance instructions.

No function to retrieve ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

Owner cant change balance

The contract owner does not have the authority to modify the balance of tokens at other addresses.

Honeypot Risk

This does not appear to be a honeypot

We are not aware of any code that prevents the sale of tokens.

No trading cooldown

The token contract has no trading cooldown function.If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

No Anti Whale

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

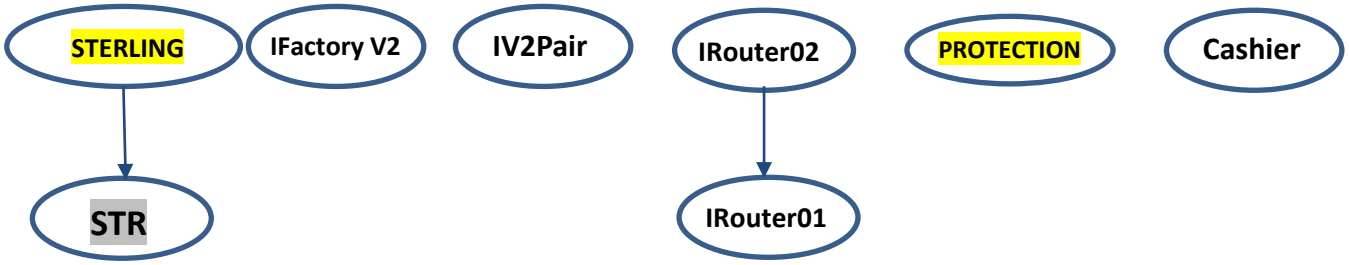
No blacklist function

No blacklist function is included.

No whitelist function

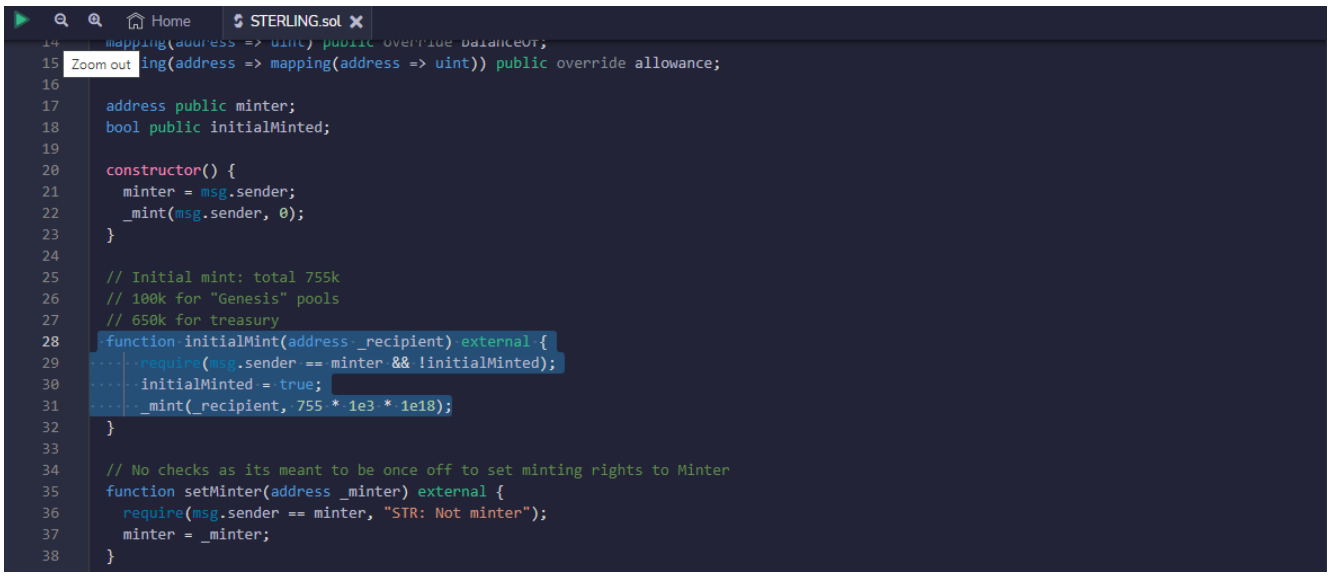
Whitelist function found

INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of STERLING	Medium #

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



MANUAL REVIEW

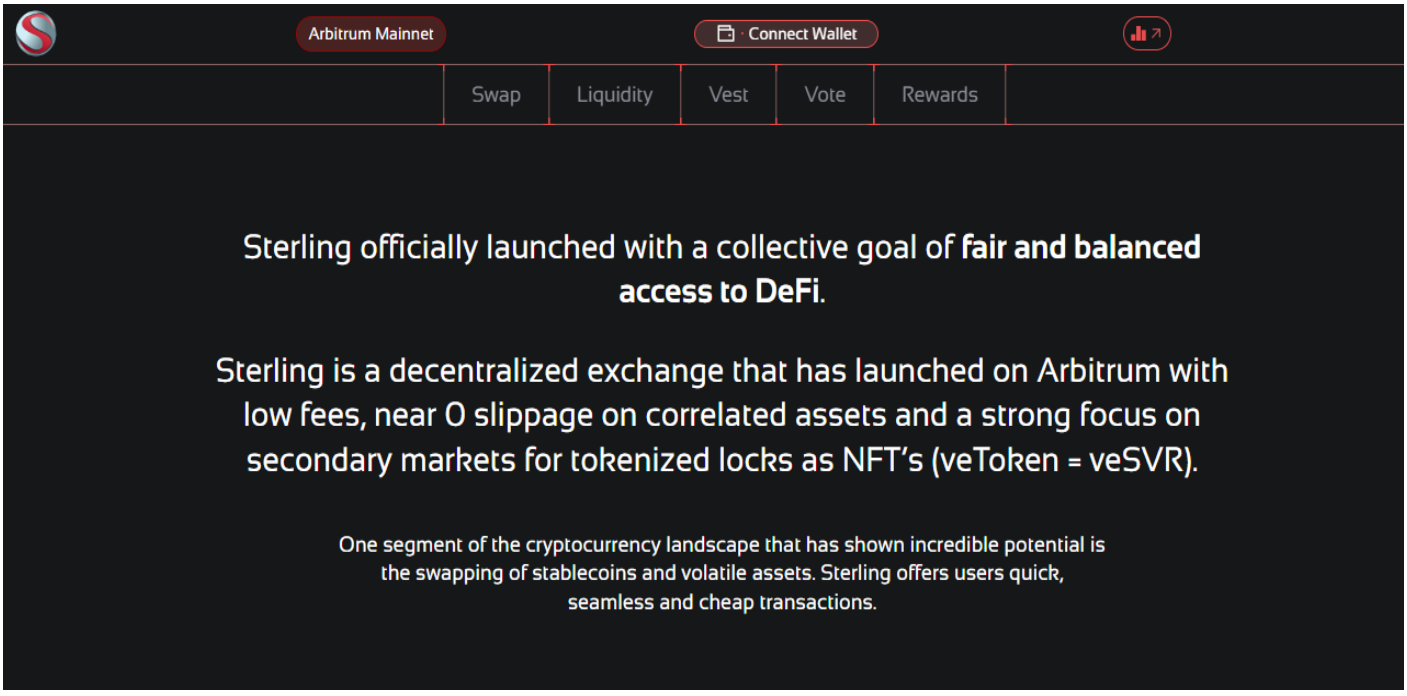
STARLING: Sterling Finance aims to act as a solution for protocols on Arbitrum to properly incentivize liquidity for their own use cases. Building on top of the groundwork laid out by Solidly, our team has addressed that first iteration's core issues to realize its full potential.

TOKEN NAME: STERLING
Ticker: STR

Chain/Standard: ARBITRUM

Total Supply: 1,005,000

The STERLING Platform Is Launched On Arbitrum



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 solidity 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 solidity 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 approve(address _spender, uint _value) external override returns (bool) {
  require(_spender != address(0), "STR: Approve to the zero address");
  allowance[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
}
```

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 ARBPIG 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.

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 OTHER WEBSITES

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

Vital Block Dedicated to securing Public and Private Blockchain Ecosystem