



# SOLIDProof

*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

MADE IN GERMANY

# Bullfarm

# Audit

**Security Assessment  
10. October, 2022**

**For**



**SolidProof\_io**



**@solidproof\_io**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	24
Source Units in Scope	27
Critical issues	28
High issues	28
Medium issues	28
Low issues	28
Informational issues	29
Audit Comments	29
SWC Attacks	30

# Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Uniswap, Uniswap, PancakeSwap etc’...)

**SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.**

SolidProof.io Reports represent an extensive auditing 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. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	10. October 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Ethereum (ERC20)

## **Website**

<https://ushi.pro/>

<https://eth-bull.app/>

## **Telegram**

[https://t.me/u\\_s\\_h\\_i](https://t.me/u_s_h_i)

## **Twitter**

[https://twitter.com/USHI\\_ETH](https://twitter.com/USHI_ETH)

## **Instagram**

[https://instagram.com/ushi\\_eth?igshid=YmMyMTA2M2Y=](https://instagram.com/ushi_eth?igshid=YmMyMTA2M2Y=)

## **Youtube**

<https://youtube.com/channel/UCFROBrqKsVzLB3UTPtZxAGA>

## Description

Bull is a faithful assistant that designed to help to taste the sweetness of a big growth

## Project Engagement

During the 7th of October 2022, **Bullfarm Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Link

**v1.0**

• TBA

# Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
<b>Critical</b>	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
<b>High</b>	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
<b>Medium</b>	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
<b>Low</b>	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
<b>Informational</b>	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

# Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## **Methodology**

The auditing process follows a routine series of steps:

1. Code review that includes the following:
  - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
  - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
  - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
  - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	2



## Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

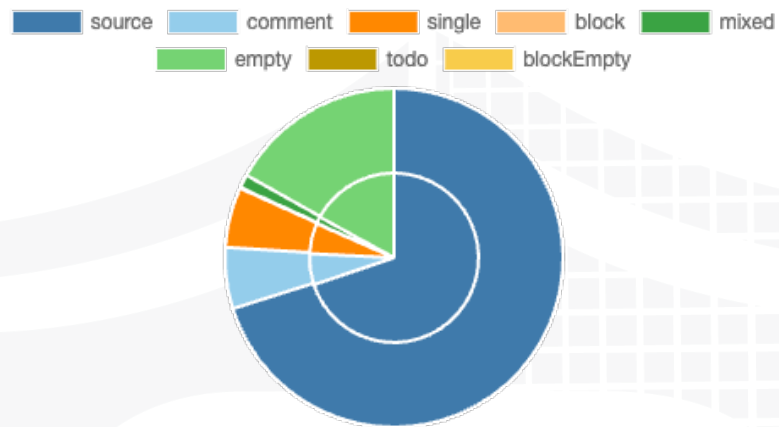
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

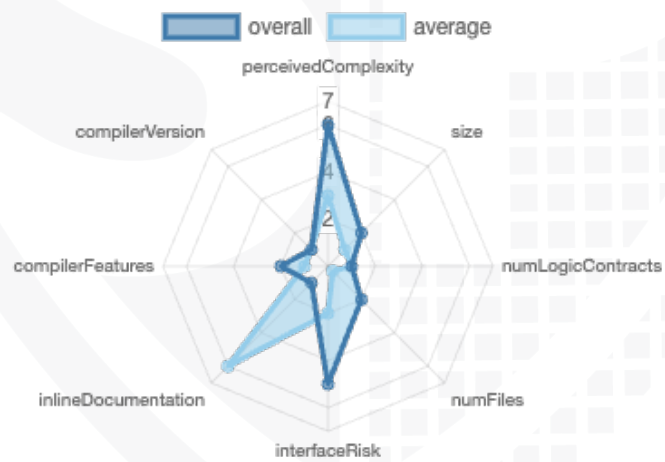
File Name	SHA-1 Hash
contracts/interfaces/IBullFund.sol	b0c205510e2d1124192750f1d1f90c7f64bcac5c
contracts/interfaces/IUshiToken.sol	41f9efb66aa2bb750e083db1b5ea40ebf4912cf7
contracts/interfaces/IBullFarm.sol	544f2468bd43c45120ec9fa46df5efbe119fd434
contracts/BullFarm.sol	2f21856259667b82f0c1e96df00b6835af94c714
contracts/UshiToken.sol	0c2704f4dfda778094cacdf927b4725e5307002b
contracts/LineManager.sol	a7a1376f5f9c79ea21d316a8b5496bfa28afe000
contracts/BullFarmFund.sol	1f54f988657a44412d22ac4d8e31a38f5dba3927

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	0	3	0

### Exposed Functions

*This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.*

Version	Public	Payable
1.0	73	8

Version	External	Internal	Private	Pure	View
1.0	39	61	28	3	33

### State Variables

Version	Total	Public
1.0	73	49

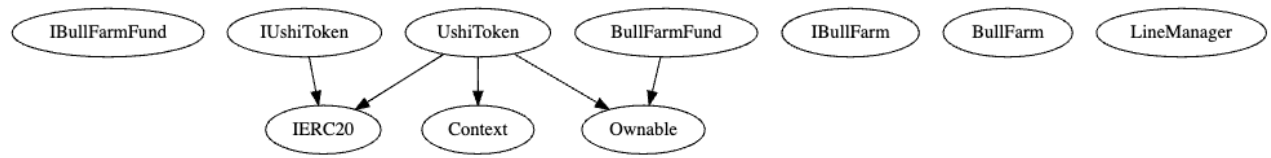
### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.8.0</code>		yes		

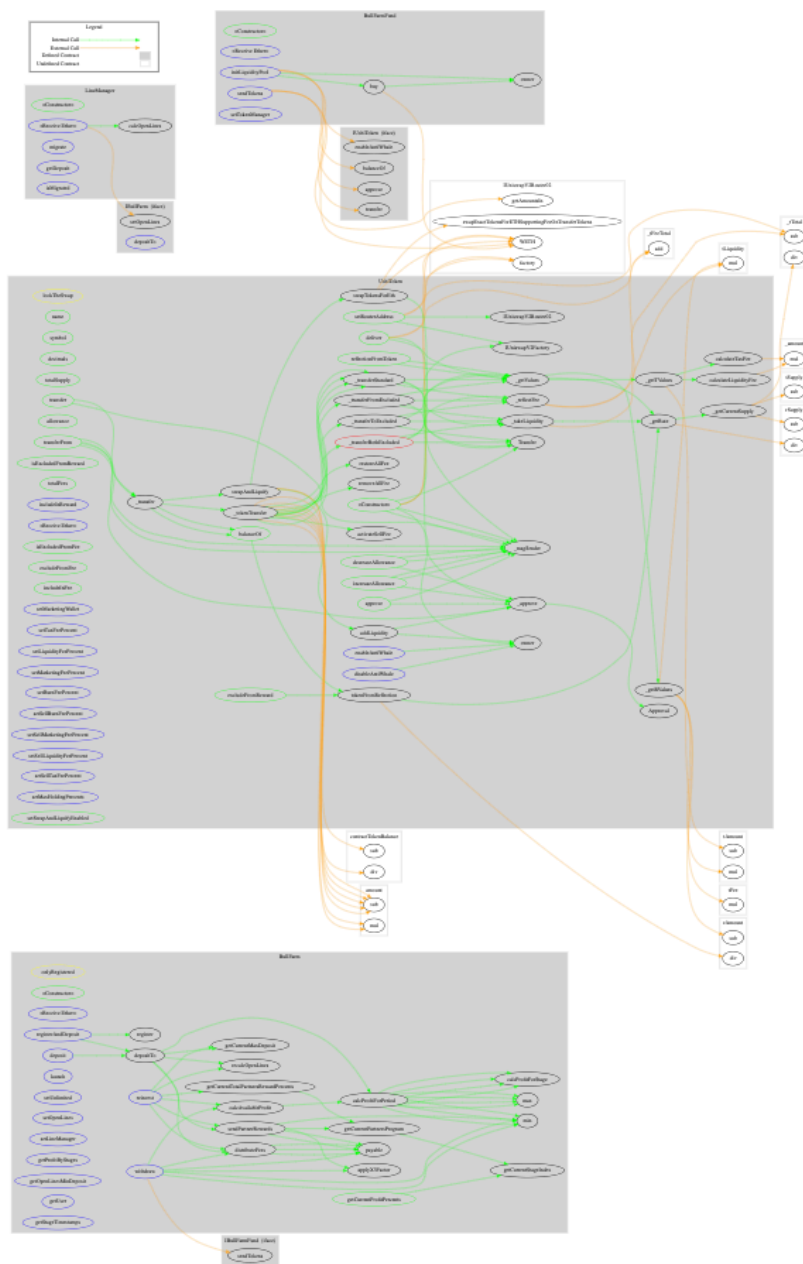
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes					

# Inheritance Graph

## v1.0



# CallGraph v1.0



## Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)



## Is contract an upgradeable

Name	
Is contract an upgradeable?	No



## Correct implementation of Token standard

ERC20				
Function	Description	Exist	Tested	Verified
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓



## Write functions of contract v1.0

<div>▼ BULLFARMFUND</div> <div>buy</div> <div>initLiquidityPool</div> <div>renounceOwnership</div> <div>sendTokens</div> <div>setTokenManager</div> <div>transferOwnership</div>	<div>▼ BULLFARM</div> <div>deposit</div> <div>depositTo</div> <div>launch</div> <div>register</div> <div>registerAndDeposit</div> <div>reinvest</div> <div>setLineManager</div> <div>setOpenLines</div> <div>setUnlimited</div> <div>withdraw</div>	<div>▼ LINEMANAGER</div> <div>migrate</div>	<div>▼ USHITOKEN</div> <div>approve</div> <div>decreaseAllowance</div> <div>deliver</div> <div>disableAntiWhale</div> <div>enableAntiWhale</div> <div>excludeFromFee</div> <div>excludeFromReward</div> <div>includeInFee</div> <div>includeInReward</div> <div>increaseAllowance</div> <div>renounceOwnership</div> <div>setBurnFeePercent</div> <div>setLiquidityFeePerc...</div> <div>setMarketingFeePerc...</div> <div>setMarketingWallet</div> <div>setMaxHoldingPerc...</div> <div>setRouterAddress</div> <div>setSellBurnFeePerc...</div> <div>setSellLiquidityFeeP...</div> <div>setSellMarketingFee...</div> <div>setSellTaxFeePercent</div> <div>setSwapAndLiquifyE...</div> <div>setTaxFeePercent</div> <div>transfer</div> <div>transferFrom</div> <div>transferOwnership</div>
--	---	---	---

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	—	—	—



## Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

### v1.0

- Owner can lock user funds by
  - Setting “\_myHoldingPercent” to 0
- Tokens
  - will be burned while tx

```

416     function _tokenTransfer(address sender↑, address recipient↑, uint256 amount↑) private {
417         if (_isExcludedFromFee[sender↑] || _isExcludedFromFee[recipient↑]) {
418             removeAllFee();
419         } else if (recipient↑ == uniswapV2Pair) {
420             activateSellFee();
421         }
422     }
423     //Calculate burn amount and marketing amount
424     uint256 burnAmt = amount↑.mul(_burnFee).div(100);
425     uint256 marketingAmt = amount↑.mul(_marketingFee).div(100);
426
427     if (_isExcluded[sender↑] && !_isExcluded[recipient↑]) {
428         _transferFromExcluded(sender↑, recipient↑, (amount↑.sub(burnAmt).sub(marketingAmt)));
429     } else if (!_isExcluded[sender↑] && _isExcluded[recipient↑]) {
430         _transferToExcluded(sender↑, recipient↑, (amount↑.sub(burnAmt).sub(marketingAmt)));
431     } else if (!_isExcluded[sender↑] && !_isExcluded[recipient↑]) {
432         _transferStandard(sender↑, recipient↑, (amount↑.sub(burnAmt).sub(marketingAmt)));
433     } else if (_isExcluded[sender↑] && _isExcluded[recipient↑]) {
434         _transferBothExcluded(sender↑, recipient↑, (amount↑.sub(burnAmt).sub(marketingAmt)));
435     } else {
436         _transferStandard(sender↑, recipient↑, (amount↑.sub(burnAmt).sub(marketingAmt)));
437     }
438
439     //Temporarily remove fees to transfer to burn address and marketing wallet
440     _taxFee = 0;
441     _liquidityFee = 0;
442
443     _transferStandard(sender↑, address(0), burnAmt);
444     _transferStandard(sender↑, marketingWallet, marketingAmt);
445
446     //Restore tax and liquidity fees
447     _taxFee = _previousTaxFee;
448     _liquidityFee = _previousLiquidityFee;
449
450     if (_isExcludedFromFee[sender↑] || _isExcludedFromFee[recipient↑] || recipient↑ == uniswapV2Pair) {
451         restoreAllFee();
452     }
453 }

```

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



## Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓



## Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	—	—	—



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

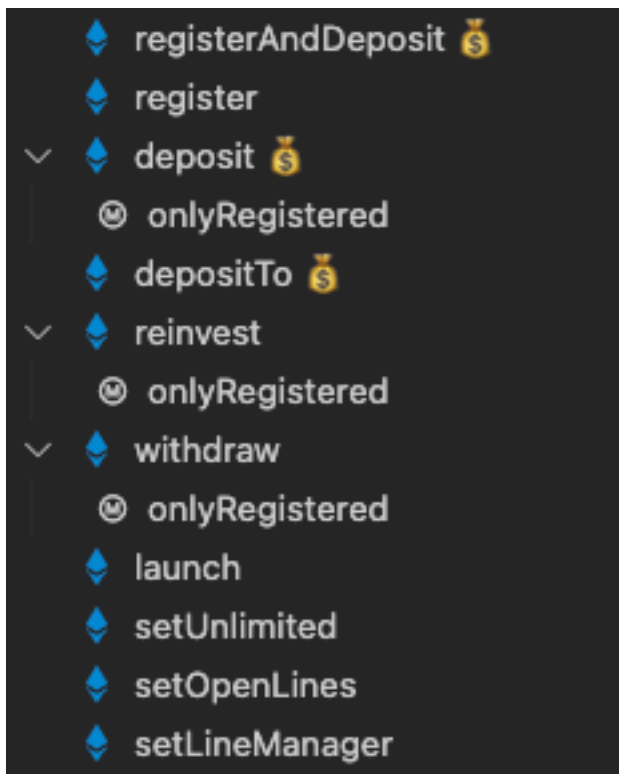
### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

# Modifiers and public functions

## v1.0

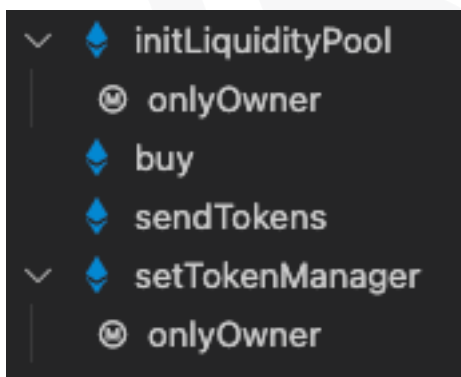
BullFarm



LineManager



BullFarmFund





## UshiToken

transfer
approve
transferFrom
increaseAllowance
decreaseAllowance
deliver
excludeFromReward
Ⓢ onlyOwner
includeInReward
Ⓢ onlyOwner
excludeFromFee
Ⓢ onlyOwner
includeInFee
Ⓢ onlyOwner
setMarketingWallet
Ⓢ onlyOwner
setTaxFeePercent
Ⓢ onlyOwner
setLiquidityFeePercent
Ⓢ onlyOwner
setMarketingFeePercent
Ⓢ onlyOwner
setBurnFeePercent
Ⓢ onlyOwner
setSellBurnFeePercent
Ⓢ onlyOwner
setSellMarketingFeePercent
Ⓢ onlyOwner
setSellLiquidityFeePercent
Ⓢ onlyOwner
setSellTaxFeePercent
Ⓢ onlyOwner
setMaxHoldingPercents
Ⓢ onlyOwner
setRouterAddress
Ⓢ onlyOwner
setSwapAndLiquifyEnabled
Ⓢ onlyOwner
enableAntiWhale
disableAntiWhale

## Comments

- Deployer can enable/disable following state variables
  - UshiToken
    - swapAndLiquifyEnabled
    - \_isExcludedFromFee
    - \_isExcluded
    - \_excluded
- Deployer can set following addresses
  - UshiToken
    - uniswapV2Pair
    - uniswapV2Router
    - marketingWallet


















- BullFarmFund
  - manager
- BullFarm
  - lineManager
- Existing Modifiers
  - onlyRegistered
  - lockTheSwap
- If an address is excluded from reward it cannot call the deliver function
- Liquidity will be added to the owner in BullFarmFund

```
39 // Add the liquidity
40 uniswapV2Router.addLiquidityETH{value: INITIAL_ETH_LIQUIDITY} (
41     address(token),
42     tokenAmount,
43     0, // slippage is unavoidable
44     0, // slippage is unavoidable
45     owner(),
46     block.timestamp
47 );
```

**Please check if an OnlyOwner or similar restrictive modifier has been forgotten.**

# Source Units in Scope

## v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/interfaces/IBullFund.sol	—————	1	8	7	3	1	3	—————
	contracts/interfaces/IUshiToken.sol	—————	1	9	8	4	1	5	—————
	contracts/interfaces/IBullFarm.sol	—————	1	9	7	3	1	8	
	contracts/BullFarm.sol	1	—————	406	406	321	26	231	 
	contracts/UshiToken.sol	1	—————	561	550	419	34	364	
	contracts/LineManager.sol	1	—————	55	55	42	1	28	
	contracts/BullFarmFund.sol	1	—————	96	96	70	9	86	 
	<b>Totals</b>	4	3	1144	1129	862	73	725	 

### Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

# Audit Results

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	3	The current pragma Solidity directive is not a certain one.
#2	BullFarm	Missing Zero Address Validation (missing-zero-check)	75-77, 231	Check that the address is not zero
#3	BullFarmFund	Missing Zero Address Validation (missing-zero-check)	94	Check that the address is not zero
#4	UshToken	Missing Zero Address Validation (missing-zero-check)	81-82,493	Check that the address is not zero
#5	UshToken	State variable visibility is not set	63	It is best practice to set the visibility of state variables explicitly
#6	LineManager	State variable visibility is not set	8-10	It is best practice to set the visibility of state variables explicitly
#7	BullFarm	State variable visibility is not set	43, 48	It is best practice to set the visibility of state variables explicitly
#8	UshToken	Local variables shadowing	318, 126	Rename the local variables that shadow another component

#9	UshIToken	Missing Events Arithmetic	513, 503, 508, 538, 518, 528, 523, 533, 498	Emit an event for critical parameter changes
#10	BullFarm	Tautology or contradiction	224	The check of "lines >= 0" is unnecessary because lines is an uint type. It cannot be below 0.

## Informational issues

Issue	File	Type	Line	Description
#1	BullFarm	State variables that could be declared constant (constable-states)	40	Add the `constant` attributes to state variables that never change
#2	UshIToken	State variables that could be declared constant (constable-states)	32, 30, 31, 26, 65	Add the `constant` attributes to state variables that never change
#3	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#4	BullFarm	Delay Withdraw	See description	We recommend you generally to add a delay of min. 24 hours to withdraw.

## Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

### 10. October 2022:

- Read whole report and modifiers section for more information

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	PASSED
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	PASSED
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	PASSED
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	PASSED
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	PASSED
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	PASSED
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	NOT PASSED
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	PASSED
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>NOT PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>



<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>NOT PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

*Solid  
Proofed*

**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

  
MADE IN GERMANY