Cyberscope

Audit Report

zkSwap Finance Farm

January 2024

Network	zkSync
Address	0x9F9D043fB77A194b4216784Eb5985c471b979D67
Audited by	© cyberscope

Table of Contents

Review

Explorer	https://explorer.zksync.io/address/0x9f9d043fb77a194b4216784eb5985c471b979d67

Audit Updates

Initial Audit	22 Jan 2024
Acknowledged Phase	24 Jan 2024

Source Files

Filename	SHA256
ZFFarm.sol	568f278e6b8799486b41b00a0c9d7ceaf66f738e3d15e32c685a5ab8bb0d0b7d
IZFToken.sol	f7e499ca5a50691f0c20d2e57c822176c62202d968b49bdb3aef6d1f95aa95da
@openzeppelin/contracts/utils/Strings.sol	0519199dbc635f98ce2e4537986604ee618bca665c65e9a1738702dfacf72010
@openzeppelin/contracts/utils/StorageSlot.sol	b4a5fb7ab93bfeda06509eafbd5f71fde0e0de84b6d9129553bd535a42166c15
@openzeppelin/contracts/utils/ShortStrings.sol	ddd52921d2996abf2e3d9c1c4f6d00194a3e3b278a164948f995862371444a55
@openzeppelin/contracts/utils/Nonces.sol	1c16c3cf8bb0679cbd47cddd8b141fea193e76966c94c858c5bccc94b8695030
@openzeppelin/contracts/utils/Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6
@openzeppelin/contracts/utils/math/SignedMath.sol	768c28e3a33c3312e57ae8a1caaec2893bc89ac6e386621de018f85e9a2d6e99

@openzeppelin/contracts/utils/math/Math.sol	a6ee779fc42e6bf01b5e6a963065706e882b016affbedfd8be19a71ea48e6e15
@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol	2fd5c641cf452efd15f784827cb2835664970d7fbc166bf80824ed27011cc374
@openzeppelin/contracts/utils/cryptography/EIP712.sol	27dac0732a0154f432c0a7a1d1f067ab51116105e157d0e5d68d040fd83954d5
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	37828cb50b47bcc51c7b770bde15d5885d871ef1e67028057a0b788c3568726e
@openzeppelin/contracts/token/ERC20/IERC20.sol	6f2faae462e286e24e091d7718575179644dc60e79936ef0c92e2d1ab3ca3cee
@openzeppelin/contracts/token/ERC20/ERC20.sol	ddff96777a834b51a08fec26c69bb6ca2d01d150a3142b3fdd8942e07921636a
@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol	912509e0e9bf74e0f8a8c92d031b5b26d2d35c6d4abf3f56251be1ea9ca946bf
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1d079c20a192a135308e99fa5515c27acfbb071e6cdb0913b13634e630865939
@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol	677cb995a34f0cc937f3d77d4626c46fbf47cdef4c9cc0314c27672c0459cf80
@openzeppelin/contracts/token/ERC20/extensions/ERC20FlashMint.sol	58f4f4e5b759b5709a7ba705dfe60a26a60fb18154bff8cdf145a7c4e8c4c368
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	2e6108a11184dd0caab3f3ef31bd15fed1bc7e4c781a55bc867ccedd8474565c
@openzeppelin/contracts/interfaces/draft-IERC6093.sol	4aea87243e6de38804bf8737bf86f750443d3b5e63dd0fd0b7ad92f77cdbd3e3
@openzeppelin/contracts/interfaces/IERC5267.sol	efd1ebd1e04b6ef9c3b8781a097588f83da954323f438d54a71dc06508e6c7b8

@openzeppelin/contracts/interfaces/IERC3156FlashLender.sol	3fb668ca6aaf756f5db9049abd2a18f638ff70307ca7ce59f85e772bae17380d
@openzeppelin/contracts/interfaces/IERC3156FlashBorrower.sol	06a759fc3607f87bfb716c95ac2f67c64b1485703bdd9467f1fea7ecc1180215
@openzeppelin/contracts/access/Ownable.sol	38578bd71c0a909840e67202db527cc6b4e6b437e0f39f0c909da32c1e30cb81

Overview

This document presents the overview of the smart contract audit conducted for the "zkSwap Finance Farm" contract. The purpose of this audit is to identify and address security vulnerabilities, provide recommendations for code improvements, and ensure the robustness of the codebase.

ZFFarm contract

This contract enables users to stake liquidity provider (LP) tokens and earn governance tokens as rewards. Key features of this contract include:

Multi-Pool Capability

It supports multiple staking pools, each associated with different LP tokens.

Dynamic Reward Allocation

The contract allocates rewards based on each pool's allocation points (allocPoint) and the total allocation points (totalAllocPoint) across all pools.

Reward Calculation

Rewards are calculated based on the time elapsed since the last reward was accounted for in each pool, using the zfPerSecond rate to determine the governance token distribution.

Customizable Reward Rate

The owner can adjust the reward rate (zfPerSecond) to modify the emission rate of governance tokens.

Staking and Withdrawal

Users can deposit LP tokens into a specific pool and withdraw them along with accrued rewards.

Emergency Withdrawal

A provision for emergency withdrawal allows users to withdraw their staked tokens without rewards, adding a safety net in unforeseen circumstances.

Governance Token Minting

The contract mints new governance tokens based on the reward rate and allocates them to stakers.

Start Time Management

The contract owner can set the start time of the farming, ensuring flexibility in the commencement of the reward generation.

Event Logging

It includes events for deposits, withdrawals, and emergency withdrawals, enhancing transparency and trackability of user actions.

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	9

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	8	1	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CCR	Contract Centralization Risk	Acknowledged
●	MC	Missing Check	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	ZFFarm.sol#L184,190
Status	Acknowledged

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

Specifically, the contract owner has the authority to make changes to key parameters of the smart contract, that can heavily impact on its functionality.

```
function updateEmissionRate(uint256 _zfPerSecond) public
onlyOwner {
    massUpdatePools();
    zfPerSecond = _zfPerSecond;
    emit EmissionRateUpdated(msg.sender, zfPerSecond,
_zfPerSecond);
}

function updateStartTime(uint256 _startTime) external onlyOwner
{
    require(startTime > block.timestamp, "Farm already
started");
    startTime = _startTime;

    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        PoolInfo storage pool = poolInfo[pid];
        pool.lastRewardTime = startTime;
    }
}
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

Team Update

The team has acknowledged that this is not a security issue and states: *Currently, the zkSwap Finance Farm contract is under a 24-hour timelock, and all information is transparent, public, and verifiable. More information can be found at https://docs.zkswap.finance/contracts-and-audits/smart-contracts*

MC - Missing Check

Criticality	Minor / Informative
Location	ZFFarm.sol#L87
Status	Unresolved

Description

The contract is processing variable `_pid` that have not been properly sanitized and checked that they form the proper shape. This variable may produce vulnerability issues.

```
function set(uint256 _pid, uint256 _allocPoint, bool
_withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
;
    poolInfo[_pid].allocPoint = _allocPoint;
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	ZFFarm.sol#L190
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function updateStartTime(uint256 _startTime) external onlyOwner
{
    require(startTime > block.timestamp, "Farm already
started");
    startTime = _startTime;

    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        PoolInfo storage pool = poolInfo[pid];
        pool.lastRewardTime = startTime;
    }
}
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	ZFFarm.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	ZFFarm.sol#L87,184,190
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function set(uint256 _pid, uint256 _allocPoint, bool
_withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
;
    poolInfo[_pid].allocPoint = _allocPoint;
}

function updateEmissionRate(uint256 _zfPerSecond) public
onlyOwner {
...
}

function updateStartTime(uint256 _startTime) external onlyOwner
{
...
}
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	ZFFarm.sol#L28
Status	Unresolved

Description

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

```
address public zf = 0x31C2c031fDc9d33e974f327Ab0d9883Eae06cA4A
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value 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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	ZFFarm.sol#L54,59,73,87,103,120,145,165,174,184,190
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256 _to
uint256 _from
address _user
uint256 _pid
IERC20 _lpToken
bool _withUpdate
uint256 _allocPoint
uint256 _amount
address _to
uint256 _zfPerSecond
uint256 _startTime
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	ZFFarm.sol#L78,91
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
totalAllocPoint = totalAllocPoint.add(_allocPoint)
totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	ZFFarm.sol#L171
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IZFToken(zf).transfer(_to, _amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ZFToken	Implementation	ERC20, Ownable		
		Public	✓	-
	mint	Public	✓	onlyMinter
	burn	Public	✓	-
	addMinter	Public	✓	onlyOwner
	removeMinter	Public	✓	onlyOwner
	getMinterLength	Public		-
	isMinter	Public		-
	getMinter	Public		-
Strings	Library			
	toString	Internal		
	toStringSigned	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	equal	Internal		
StorageSlot	Library			

	getAddressSlot	Internal		
	getBooleanSlot	Internal		
	getBytes32Slot	Internal		
	getUint256Slot	Internal		
	getStringSlot	Internal		
	getStringSlot	Internal		
	getBytesSlot	Internal		
	getBytesSlot	Internal		
ShortStrings	Library			
	toShortString	Internal		
	toString	Internal		
	byteLength	Internal		
	toShortStringWithFallback	Internal	✓	
	toStringWithFallback	Internal		
	byteLengthWithFallback	Internal		
Nonces	Implementation			
	nonces	Public		-
	_useNonce	Internal	✓	
	_useCheckedNonce	Internal	✓	
Context	Implementation			

	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
SignedMath	Library			
	max	Internal		
	min	Internal		
	average	Internal		
	abs	Internal		
Math	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	max	Internal		
	min	Internal		
	average	Internal		
	ceilDiv	Internal		
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		

	sqrt	Internal		
	log2	Internal		
	log2	Internal		
	log10	Internal		
	log10	Internal		
	log256	Internal		
	log256	Internal		
	unsignedRoundsUp	Internal		
MessageHashUtils	Library			
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toDataWithIntendedValidatorHash	Internal		
	toTypedDataHash	Internal		
EIP712	Implementation	IERC5267		
		Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		
	eip712Domain	Public		-
	_EIP712Name	Internal		
	_EIP712Version	Internal		

ECDSA	Library			
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	_throwError	Private		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
ERC20	Implementation	Context, IERC20, IERC20Metadata, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-

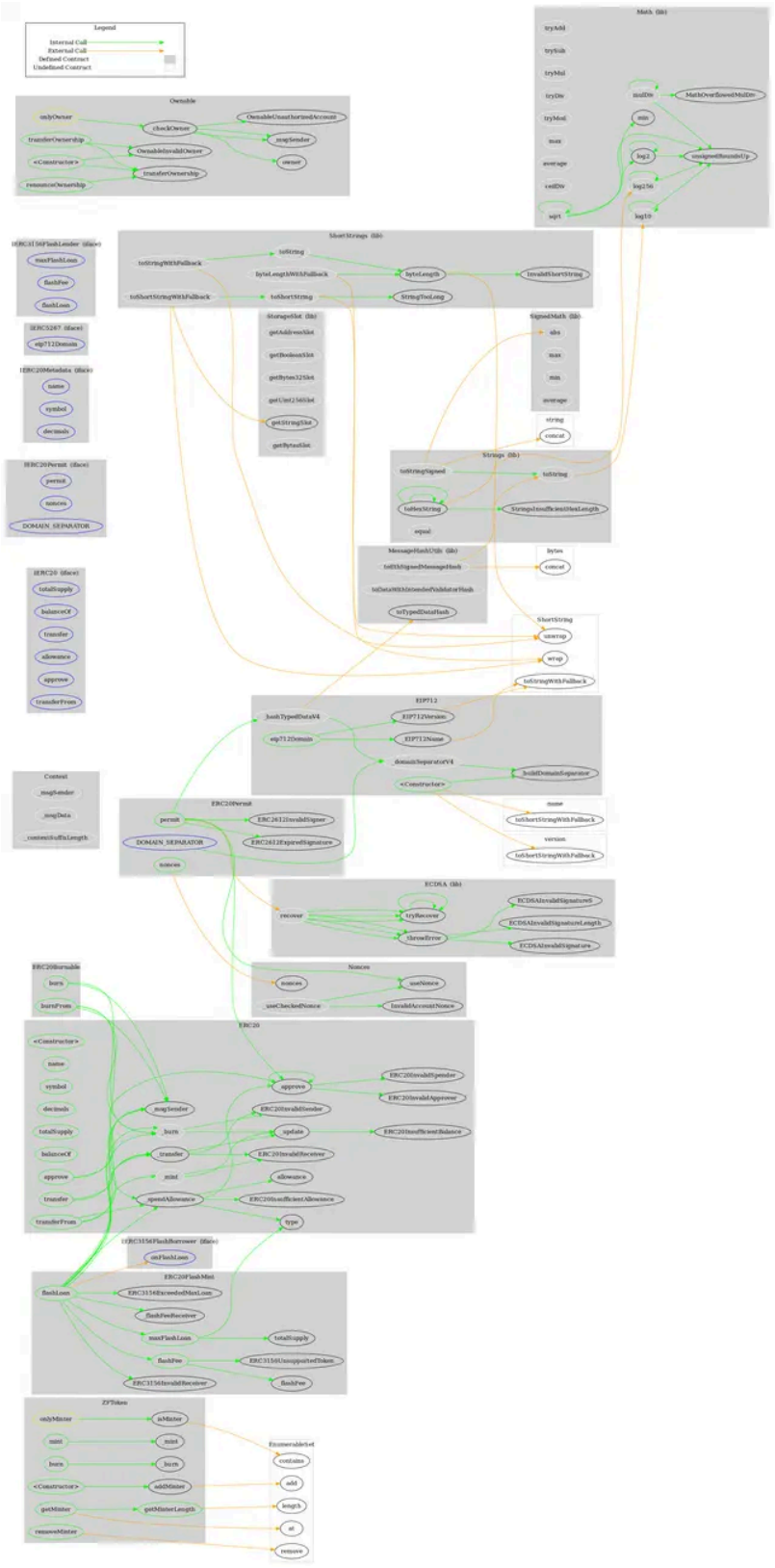
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-

	decimals	External		-
ERC20Permit	Implementation	ERC20, IERC20Permit, EIP712, Nonces		
		Public	✓	EIP712
	permit	Public	✓	-
	nonces	Public		-
	DOMAIN_SEPARATOR	External		-
ERC20FlashMint	Implementation	ERC20, IERC3156FlashLender		
	maxFlashLoan	Public		-
	flashFee	Public		-
	_flashFee	Internal		
	_flashFeeReceiver	Internal		
	flashLoan	Public	✓	-
ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
IERC20Errors	Interface			
IERC721Errors	Interface			

IERC1155Errors	Interface			
IERC5267	Interface			
	eip712Domain	External		-
IERC3156FlashLender	Interface			
	maxFlashLoan	External		-
	flashFee	External		-
	flashLoan	External	✓	-
IERC3156FlashBorrower	Interface			
	onFlashLoan	External	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

Inheritance Graph

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
checkOwner
transferOwnership
OwnableInvalidOwner
<Constructor>
transferOwnership
renounceOwnership
OwnableUnauthorizedAccount
_msgSender
owner
Math (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
max
average
ceilDiv
sqrt
mulDiv
min
log2
log256
log10
MathOverflowedMulDiv
unsignedRoundsUp
IERC3156FlashLender (iface)
maxFlashLoan
flashFee
flashLoan
IERC5267 (iface)
eip712Domain
IERC20Metadata (iface)
name
symbol
decimals
ShortStrings (lib)
toString
toStringWithFallback
byteLengthWithFallback
toShortString
toShortStringWithFallback
byteLength
StringTooLong
InvalidShortString
StorageSlot (lib)
getAddressSlot
getBooleanSlot
getBytes32Slot
getUint256Slot
getStringSlot
getBytesSlot
SignedMath (lib)
abs
max
min
average
string
concat
IERC20Permit (iface)
permit
nonces
DOMAIN_SEPARATOR
Strings (lib)
toStringSigned
toString
toHexString
equal
StringsInsufficientHexLength
MessageHashUtils (lib)
toEthSignedMessageHash
toDataWithIntendedValidatorHash
toTypedDataHash
bytes
concat
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
ShortString
unwrap
wrap
toStringWithFallback
EIP712
_hashTypedDataV4
eip712Domain
EIP712Version
EIP712Name
domainSeparatorV4
<Constructor>
buildDomainSeparator
Context
_msgSender
_msgData
_contextSuffixLength
name
toShortStringWithFallback
version
toShortStringWithFallback
ERC20Permit
permit
DOMAIN_SEPARATOR
nonces
ERC2612InvalidSigner
ERC2612ExpiredSignature
ECDSA (lib)
recover
tryRecover
throwError
ECDSAInvalidSignatureS
ECDSAInvalidSignatureLength
ECDSAInvalidSignature
ERC20Burnable
burn
burnFrom
Nonces
nonces
useNonce
useCheckedNonce
InvalidAccountNonce
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_msgSender
burn
transfer
mint
spendAllowance
approve
update
allowance
type
ERC20InvalidSpender
ERC20InvalidApprover
ERC20InvalidSender
ERC20InsufficientBalance
ERC20InvalidReceiver
ERC20InsufficientAllowance
IERC3156FlashBorrower (iface)
onFlashLoan
ERC20FlashMint
flashLoan
ERC3156ExceededMaxLoan
flashFeeReceiver
maxFlashLoan
flashFee
ERC3156InvalidReceiver
totalSupply
ERC3156UnsupportedToken
flashFee
ZFToken
onlyMinter
isMinter
mint
burn
<Constructor>
addMinter
getMinter
getMinterLength
removeMinter
EnumerableSet
contains
add
length
at
remove

Summary

zkSwap Finance Farm contract implements a financial and rewards mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. 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 Cyberscope's prior written consent. 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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. 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 Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io