Cyberscope

Audit Report

TG.Bet Staking

January 2024

SHA256 5522169c507e0d552a5b1f8909c0cdd34a312757a5702405eb000f3ce104aee9

Audited by © cyberscope

Table of Contents

Review

Testing Deploy	https://testnet.bscscan.com/address/0xe3433db02415063eb1baf0119cc90c7c8a126278

Audit Updates

Initial Audit	18 Jan 2024

Source Files

Filename	SHA256
contracts/StakingManagerV1.sol	5522169c507e0d552a5b1f8909c0cdd34a312757a5702405eb000f3ce104aee9
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	2d3d7dc6e116cb8ebb8517208141cb3d0950b337a285f15f8476ec3df29d824e
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	db92fc1b515decad3a783b1422190877d2d70b907c6e36fb0998d9465aee42db
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	78a6bc84bbb417f0d8a6b12e181e0f783151774f4f0c054c5d3f920e70d69f8c
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	53e882762222514d0f4ffe2c0c28fd59709987197ba11f0f3a8f35b72fd33bb6
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20PermitUpgradeable.sol	d04c39cbfbc33e17a6ec68df23f0ad97f8830330bcfd4b0be74b1d1ce6101600
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	a2c4e5c274a586f145d278293ae33198cd8f412ab7e6d26f2394c8949b32b24b
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	2d9e57d2a4b0775334be2968019c1939377d45b69e8b724fe6bb80af47e28419

Overview

The Staking Manager contract is designed to facilitate staking functionalities, providing users with the ability to deposit tokens, harvest rewards, and withdraw funds. Key features include:

Initialization

The contract is initialized with parameters such as the reward token address, reward distribution rate per block, lock time, and the end block for staking.

Deposit Functionality

Users can deposit tokens into the staking pool. A dedicated deposit function for the presale contract is available, allowing presale participants to deposit tokens seamlessly.

Withdrawal with Lock Period

Users are allowed to withdraw their staked tokens, subject to a lock period. The lock period ensures that users cannot withdraw before a specified time, promoting the stability and integrity of the staking platform.

Reward Harvesting

Users can harvest their staked rewards, which are accumulated based on the time and amount staked. Rewards are distributed per block, and the contract maintains precise calculations for accurate reward distribution.

Reward Calculation

The contract calculates rewards based on the number of blocks since the last reward distribution. Accurate reward distribution is ensured through meticulous tracking of the accumulated rewards per share.

Presale Integration

Presale participants benefit from a specialized deposit function, streamlining the interaction between presale and staking functionalities.

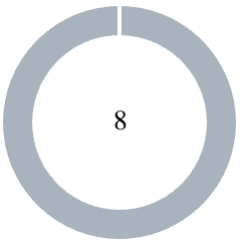
Ownership and Configuration Management

The contract includes functions allowing the owner to set parameters such as the presale contract address, staked token address, lock time, end block, launch time, and reward distribution rate.

Presale-Only Functionality

Certain functions are restricted to the presale contract to maintain control and security.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	8

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	CR	Code Repetition	Unresolved
	CCR	Contract Centralization Risk	Unresolved
	LPCL	Locking Period Calculation Logic	Unresolved
	TSI	Tokens Sufficiency Insurance	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L06	Missing Events Access Control	Unresolved
	L07	Missing Events Arithmetic	Unresolved
	L16	Validate Variable Setters	Unresolved

CR - Code Repetition

Criticality	Minor / Informative
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
(staker.amount * accumulatedRewardsPerShare) / REWARDS_PRECISION;
...
function deposit(uint256 _amount) external {}
...
function depositByPresale(address _user,uint256 _amount) external
onlyPresale {}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L230
Status	Unresolved

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.

- setPresale()
- setStakeToken()
- setLockTime()
- setEndBlock()
- setLaunchTime()
- setRewardsPerBlock()

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.

LPCL - Locking Period Calculation Logic

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L134
Status	Unresolved

Description

The current logic of the withdraw method checks if the staking occurred after the contract launch and, if so, compares the expiration time with the current block timestamp. This calculation of unlockTime is resulting in a doubling of the waiting time for users who participate in the presale and stake tokens right before it ends.

Upon closer analysis, the condition `unlockTime + lockTime <= block.timestamp` implies that the locking period is multiplied by two, leading to an extended withdrawal delay for certain users. This may create an unfair advantage for those who staked tokens during specific periods, potentially affecting the user experience and fairness of the staking platform.

```
The condition could be analyzed as the following:

uint256 unlockTime = staker.stakedTime + lockTime;

unlockTime + lockTime <= block.timestamp ->
(staker.stakedTime + lockTime + lockTime) <= block.timestamp, ->
(staker.stakedTime + lockTime*2) <= block.timestamp, ->
```

```
uint256 unlockTime = staker.stakedTime + lockTime;
if (staker.stakedTime >= launchTime) {
    require(
        unlockTime <= block.timestamp,
        "You are not allowed to withdraw before locked time"
    );
} else if (unlockTime >= launchTime) {
    require(
        unlockTime + lockTime <= block.timestamp,
        "You are not allowed to withdraw before locked time"
    );
} else {
    require(
        launchTime + lockTime <= block.timestamp,
        "You are not allowed to withdraw before locked time"
    );
}
```

Recommendation

Review the locking period calculation to ensure a consistent and fair withdrawal experience for all users, irrespective of the staking timing during the presale.

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Status	Unresolved

Description

The tokens are not held within the contract itself. Instead, the contract is designed to provide the tokens from an external administrator. While external administration can provide flexibility, it introduces a dependency on the administrator's actions, which can lead to various issues and centralization risks.

```
stakeToken.safeTransfer(_user, rewardsToHarvest);
```

Recommendation

It is recommended to consider implementing a more decentralized and automated approach for handling the contract tokens. One possible solution is to hold the presale tokens within the contract itself. If the contract guarantees the process it can enhance its reliability, security, and participant trust, ultimately leading to a more successful and efficient process.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L47,48,49,50,71,100,101,215,230,234,238,242,246,251
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.

```
address _rewardTokenAddress
uint256 _rewardTokensPerBlock
uint256 _lockTime
uint256 _endBlock
uint256 _amount
address _user
address _presale
address _stakeToken
uint256 _launchTime
```

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.

L06 - Missing Events Access Control

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L231
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. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
presaleContract = _presale
```

Recommendation

To avoid this issue, 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.

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L243,253
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.

```
endBlock = _endBlock
rewardTokensPerBlock = _rewardTokensPerBlock
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/StakingManagerV1.sol#L231
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
presaleContract = _presale
```

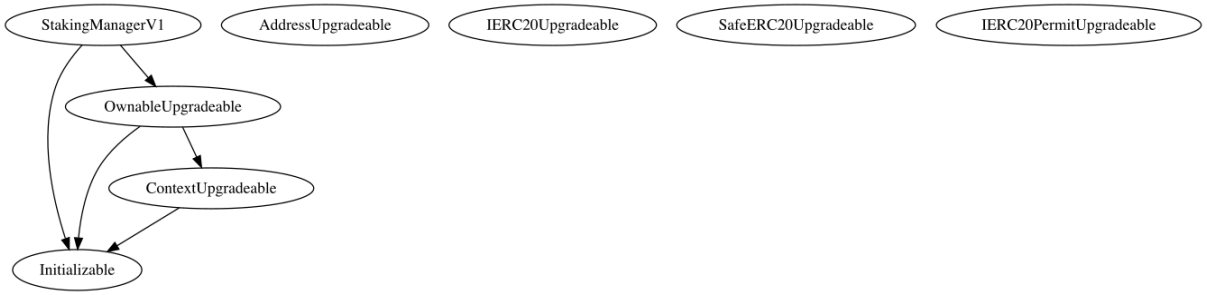
Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

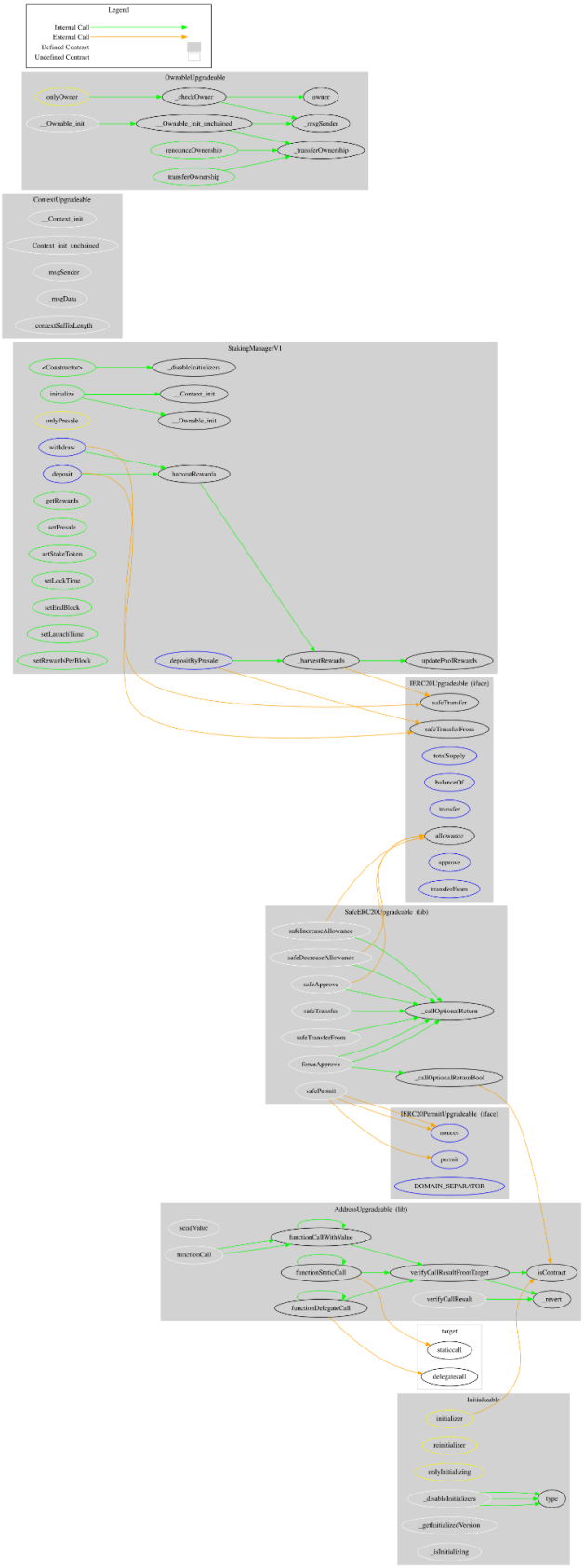
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
StakingManagerV1	Implementation	Initializable, OwnableUpgradeable		
		Public	✓	-
	initialize	Public	✓	initializer
	deposit	External	✓	-
	depositByPresale	External	✓	onlyPresale
	withdraw	External	✓	-
	harvestRewards	Public	✓	-
	_harvestRewards	Private	✓	
	updatePoolRewards	Private	✓	
	getRewards	Public		-
	setPresale	Public	✓	onlyOwner
	setStakeToken	Public	✓	onlyOwner
	setLockTime	Public	✓	onlyOwner
	setEndBlock	Public	✓	onlyOwner
	setLaunchTime	Public	✓	onlyOwner
	setRewardsPerBlock	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
OwnableUpgradeable
onlyOwner
_checkOwner
owner
__Ownable_init
__Ownable_init_unchained
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
ContextUpgradeable
__Context_init
__Context_init_unchained
_msgSender
_msgData
_contextSuffixLength
StakingManagerV1
<Constructor>
_disableInitializers
initialize
Context_init
__Ownable_init
onlyPresale
withdraw
deposit
harvestRewards
getRewards
setPresale
setStakeToken
setLockTime
setEndBlock
setLaunchTime
setRewardsPerBlock
depositByPresale
_harvestRewards
updatePoolRewards
IERC20Upgradeable (iface)
safeTransfer
safeTransferFrom
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeERC20Upgradeable (lib)
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
safeTransfer
_callOptionalReturn
safeTransferFrom
forceApprove
_callOptionalReturnBool
safePermit
IERC20PermitUpgradeable (iface)
nonces
permit
DOMAIN_SEPARATOR
AddressUpgradeable (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
verifyCallResultFromTarget
verifyCallResult
isContract
revert
target
staticcall
delegatecall
Initializable
initializer
reinitializer
onlyInitializing
_disableInitializers
type
_getInitializedVersion
_isInitializing

Summary

The Staking Manager V1 contract enables users to stake and harvest rewards. It features deposit functions for regular users and a specialized one for presale participants. Users can withdraw after a lock period, ensuring staking stability. The contract calculates rewards per block, tracks staking metrics, and integrates with presale processes. Ownership functions allow for easy configuration adjustments. Upgradeability is built in for future enhancements. 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