Cyberscope

Audit Report

ADstaking

December 2022

SHA256 498cd13885fd52cb0dee96c67c5278c4185ea671c18d2d737d6594ccf4c66c99

Audited by © cyberscope

Table of Contents

Contract Review

Contract Name	ADStaking
Compiler Version	v0.6.12+commit.27d51765
Optimization	200 runs
Licence	None
Explorer	https://bscscan.com/token/0x253A8bdaca140F154FA63bCbd89e249F4611eB1F

Source Files

Filename	**SHA256**
contract.sol	498cd13885fd52cb0dee96c67c5278c4185ea671c18d2d737d6594ccf4c66c99

Audit Updates

Initial Audit	6th December 2022
Corrected	

Introduction

The ADstaking contract implements a staking contract.

Roles

The contract has an owner role.

Owner Role

The owner has the authority to

- Add a new pool.
- Configure pool parameters.
- Enable or disable the emergency withdrawal mechanism.
- Add balance for the treasury.
- Recover treasury tokens when the farming time elapses.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CRI	Claim Reward Inconsistency	Unresolved
●	ADF	Accidentally Deposited Funds	Unresolved
●	STC	Succeeded Transfer Check	Unresolved
●	MC	Missing Check	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

CRI - Claim Reward Inconsistency

Criticality	minor / informative
Location	contract.sol#L1067
Status	Unresolved

Description

The contract distributes the rewarded tokens even if the amount is insufficient. This may produce issues since the users would receive a different amount than expected.

```
function safeTokenTransfer(address _to, uint256 _amount) internal {
        uint256 tokenBal = token.balanceOf(address(this));
        if (_amount > tokenBal) {
            token.transfer(_to, tokenBal);
        } else {
            token.transfer(_to, _amount);
        }
}
```

Recommendation

The team could follow two alternatives:

- Revert the transaction if the reward amount is insufficient.
- If reverting the deposit/withdraw is not desirable, the team could move the claim rewards functionality to a different method.

ADF - Accidentally Deposited Funds

Criticality	minor / informative
Location	contract.sol#L1043
Status	Unresolved

Description

The contract might receive tokens from an external source. If someone deposits tokens to the contract it could be counted as treasure.

```
function addBalance(uint256 _amount, uint256 _endBlock) external onlyOwner {
        require( _amount > 0 , "Cant add 0 tokens");
        require( _endBlock > block.number , "end block should be greater than 
current block");

        uint256 oldBalance = token.balanceOf(address(this));
        token.safeTransferFrom(msg.sender, address(this), _amount);
        uint256 newBalance = token.balanceOf(address(this));
        _amount = newBalance.sub(oldBalance);

        endBlock = _endBlock;
        treasure = treasure.add(_amount);
        blocks = _endBlock - block.number;
        tokenPerBlock = treasure.div(blocks);
        startBlock = block.number;
}
```

Recommendation

The team could take into consideration `token.balanceOf(address(this))` on the treasure aggragation. Hence, the treasury could be the `amount + (token.balanceOf(address(this)) - sum of pool balances)`

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract.sol#L1067,1080
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.

```
function safeTokenTransfer(address _to, uint256 _amount) internal {
        uint256 tokenBal = token.balanceOf(address(this));
        if (_amount > tokenBal) {
            token.transfer(_to, tokenBal);
        } else {
            token.transfer(_to, _amount);
        }
}

function recoverTreasure( IBEP20 recoverToken, uint256 amount) external
onlyOwner {
        require(recoverToken != token,"Cant withdraw native token");
        require(block.number > endBlock, "can recover only farming end.");
        recoverToken.transfer(msg.sender, amount);
}
```

Recommendation

The contract should check if the result of the transfer methods is successful.

MC - Missing Check

Criticality	minor / informative
Location	contract.sol#L828
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

The constructor arguments have not been properly sanitized.

```
constructor(
    IBEP20 _token,
    uint256 _startBlock,
    uint256 _tokenPerBlock
) public {
    token = _token;
    startBlock = _startBlock;
    tokenPerBlock = _tokenPerBlock;
    token.balanceOf( address(this) );
}
```

The `set`method can arbitrary initiate any pool even if it does not exist.

```
function set(uint256 _pid, uint256 _allocPoint, uint256 _harvestInterval, bool
_withUpdate,
        uint256 _withdrawLockPeriod ) external onlyOwner {
        require(_withdrawLockPeriod <= 90 days, "withdraw lock must be less
than 90 days");
        require(_harvestInterval <= 90 days, "set: invalid harvest interval");
        if (_withUpdate) {
            massUpdatePools();
        }
        ..
}
```

Recommendation

The contract should properly check the variables according to the required specifications.

- The variable _startBlock should be greater than the current timestamp.
- The variable _tokenPerBlock should be greater than zero.
- The function set could check if the corresponding pool id exits.
- The set function should validate if the corresponding _pid exists.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract.sol#L1039,864,901,956,1067,878,933,976,1000,915,844,886,865,986,1004,845
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_endBlock
_harvestInterval
_pid
_amount
_allocPoint
_from
_user
_status
_withUpdate
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L07 - Missing Events Arithmetic

Criticality	minor / informative
Location	contract.sol#L864,1039,844
Status	Unresolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

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

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor / informative
Location	contract.sol#L580,371,422,481,471,447,571,555,457,397
Status	Unresolved

Description

Functions that are not used in the contract, and make the code's size bigger.

```
safeDecreaseAllowance
sendValue
functionCallWithValue
functionDelegateCall
functionStaticCall
safeIncreaseAllowance
safeApprove
functionCall
...
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contract.sol#L956
Status	Unresolved

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(bool,string)(isLocked == false,withdraw still locked)
```

Recommendation

Remove the equality to the boolean constant.

L15 - Local Scope Variable Shadowing

Criticality	minor / informative
Location	contract.sol#L1000
Status	Unresolved

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
_status
```

Recommendation

The local variables should have different names from the upper scoped variables.

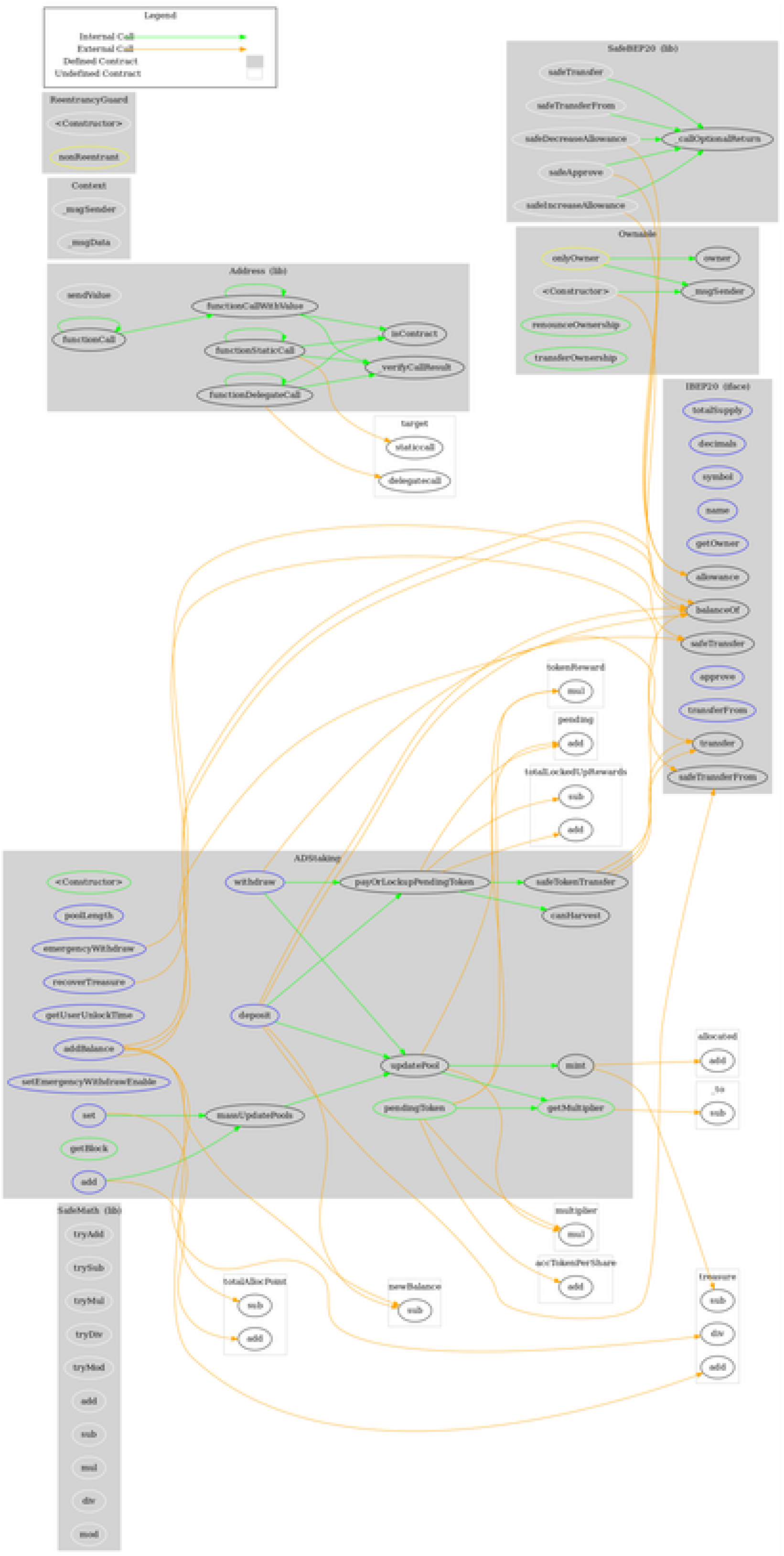
Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Address	Library			
	isContract	Internal		

	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	functionDelegateCall	Internal	✓	
	_verifyCallResult	Private		
SafeBEP20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	_callOptionalReturn	Private	✓	
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
ReentrancyGuard	Implementation			
	<Constructor>	Internal	✓	
ADStaking	Implementation	Ownable, ReentrancyGuard		

	<Constructor>	Public	✓	-
	poolLength	External		-
	add	External	✓	onlyOwner
	set	External	✓	onlyOwner
	getMultiplier	Public		-
	pendingToken	Public		-
	canHarvest	Public		-
	massUpdatePools	Public	✓	-
	updatePool	Public	✓	-
	deposit	External	✓	nonReentrant
	withdraw	External	✓	nonReentrant
	getUserUnlockTime	External		-
	emergencyWithdraw	External	✓	nonReentrant
	setEmergencyWithdrawEnable	External	✓	onlyOwner
	payOrLockupPendingToken	Internal	✓	
	addBalance	External	✓	onlyOwner
	mint	Internal	✓	
	safeTokenTransfer	Internal	✓	
	getBlock	Public		-
	recoverTreasure	External	✓	onlyOwner

Contract Flow



Summary

ADstaking implements a staking mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

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