Cyberscope

Audit Report

Smartrade

October 2023

GitHub	https://github.com/SmarTradeDev/DepositContract
Commit	99acc0efd710fc5248be41788fe1373c77ae7f52
Audited by	© cyberscope

Table of Contents

Review

Repository	https://github.com/SmarTradeDev/DepositContract
Commit	99acc0efd710fc5248be41788fe1373c77ae7f52
Testing Deploy	https://testnet.bscscan.com/address/0xefeece878c360d622bbb2f48ee4369790efb49be

Audit Updates

Initial Audit	12 Oct 2023 https://github.com/cyberscope-io/audits/blob/main/smartrade/v1/audit.pdf
Corrected Phase 2	14 Oct 2023

Source Files

Filename	SHA256
contracts/deposit.sol	8addd2be14b751ca46c0f9026d655073ed90bfa4766f6d37294e354b1bcfd034

Overview

Cyberscope audited the SmarTradeContract contract from the SmartTrade ecosystem. Its primary purpose is to manage staking of a specific token and provide rewards to users. It supports various features, including adding and editing staking packages with specified amounts, durations, and interest rates. Users can stake their tokens into these packages, specifying a referrer, and receive rewards based on the staking package's rate. The contract allows users to claim their rewards and provides a mechanism to unstake tokens. Additionally, the contract is designed with an owner who can modify package details, precision, and the referrer rate. It also includes some security measures to prevent contract calls from other contracts, ensuring that staking and claiming are primarily for externally owned accounts (EOAs). The contract's core logic revolves around staking, tracking rewards, and enabling users to interact with these features securely.

Roles

Owner

The owner has the authority to:

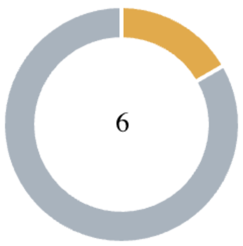
- `Set the precision of the contract.`
- `Set the referrer rate.`
- `Add and edit staking packages.`
- `Renew the owner's address.`
- `Claim contract tokens.`

User

The user has the authority to:

- `Stake tokens into specific packages.`
- `Claim rewards.`
- `Unstake their tokens.`

Findings Breakdown



●	Critical	0
●	Medium	1
●	Minor / Informative	5

	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	1	0	0
●	Minor / Informative	2	3	0	0

Diagnostics

🔴 Critical 🟠 Medium ⚪ Minor / Informative

Severity	Code	Description	Status
🟠	MEE	Missing Events Emission	Acknowledged
⚪	RRC	Redundant Require Condition	Unresolved
⚪	MEM	Misleading Error Messages	Unresolved
⚪	PTAI	Potential Transfer Amount Inconsistency	Acknowledged
⚪	TSI	Tokens Sufficiency Insurance	Acknowledged
⚪	OCTD	Transfers Contract's Tokens	Acknowledged

MEE - Missing Events Emission

Criticality	Medium
Location	contracts/deposit.sol#L199
Status	Acknowledged

Description

The `unStake` function emit the `Staked` event instead of the `UnStaked`. 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 unStake(uint256 stakingId) public onlyEOA onlyStaker(stakingId) {
        ...
        emit Staked(msg.sender, staking.stakeToken, staking.packageIdx);
    }
```

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. Additionally it is recommended to emit the `UnStaked` event in the `unStake` function.

RRC - Redundant Require Condition

Criticality	Minor / Informative
Location	contracts/deposit.sol#L57
Status	Unresolved

Description

Since the `msg.sender` is indeed the staker of the staking at index `stakingIdx` , it implies that `stakingIdx` is a valid index within the stakings array. This is because the staking at index `stakingIdx` was created by the current `msg.sender` , and it is not possible for `stakingIdx` to exceed the length of the stakings array in this context.

As a result, the first statement, `require(stakingIdx < stakings.length, "Invalid staking index");` , is redundant and can be safely removed without impacting the security or functionality of the contract. Removing this redundant condition can lead to cleaner and more efficient code.

```
require(stakingIdx < stakings.length, "Invalid staking index");
require(msg.sender == stakings[stakingIdx].staker, "Not staker");
```

Recommendation

We recommend removing the redundant condition `require(stakingIdx < stakings.length, "Invalid staking index");` from the codebase to enhance its readability and efficiency without compromising the security of the smart contract.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	contracts/deposit.sol#L64,145,177,180,192,213
Status	Unresolved

Description

The contract is not ussing error messages in revert statements. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(newOwner != address(0));
require(claimRes);
require(referRewardRes);
require(unstakeRes);
require(transferTokenRes);
require(depositRes);
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	contracts/deposit.sol#L131,144
Status	Acknowledged

Description

The `transfer()` and `transferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
package.amount * (10 ** IToken(tokenAddr).decimals())
...
IToken(newStaking.stakeToken).transferFrom(newStaking.staker, address(this),
newStaking.stakeAmount);
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts. Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

```
Actual Transferred Amount = Balance After Transfer - Balance Before Transfer
```

Team Update

The team has acknowledged that this is not a security issue and states:

We currently support only USDT, USDC and DAI for staking, and all these tokens have zero fee transfer. So, I think no more mechanism to fix this would be needed.

TSI - Tokens Sufficiency Insurance

Criticality	Minor / Informative
Location	contracts/deposit.sol#L176,179,191
Status	Acknowledged

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.

```
bool claimRes = IToken(staking.stakeToken).transfer(staking.staker, rewards);
        require(claimRes);
 if(staking.closed) {
            bool referRewardRes =
IToken(staking.stakeToken).transfer(staking.referrer, (staking.claimedAmount -
staking.stakeAmount) * referrerRate / precision);
            require(referRewardRes);
...
 if(staking.claimedAmount < staking.stakeAmount) {
            bool unstakeRes =
IToken(staking.stakeToken).transfer(staking.staker, staking.stakeAmount -
staking.claimedAmount);
            require(unstakeRes);
```

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.

Team Update

The team has acknowledged that this is not a security issue and states:

Due to the platform mechanism, the daily income would be sent to the contract regularly.

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	contracts/deposit.sol#L202
Status	Acknowledged

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `depositToVault` function.

```
function depositToVault(address token, address to, uint256 amount) public
onlyOwner {
    IToken(token).transfer(to, amount);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

Team Update

The team has acknowledged that this is not a security issue and states:

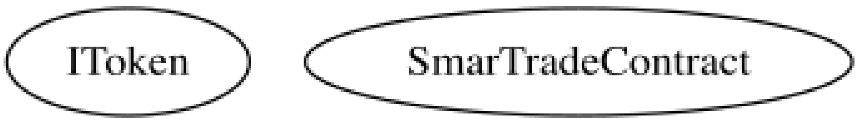
We will use a multisig wallet for powerful security mechanism.

Functions Analysis

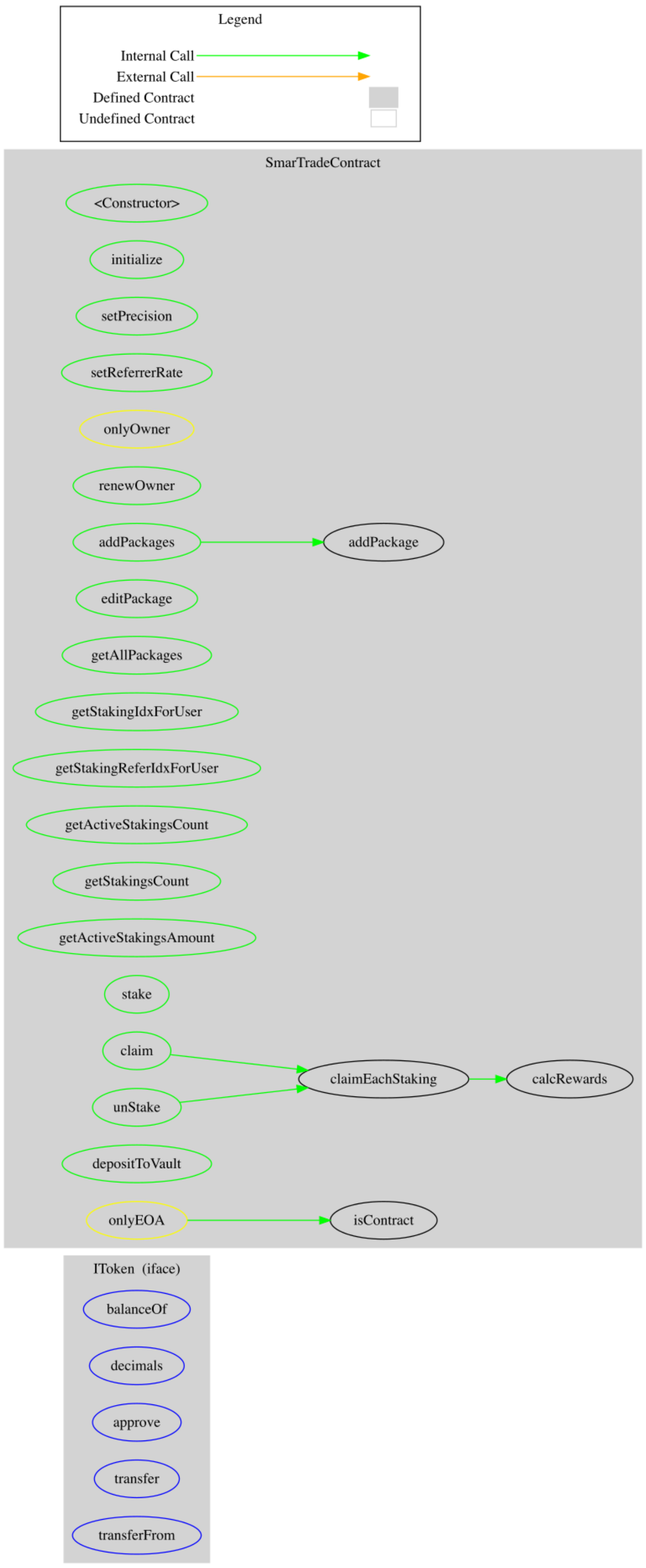
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IToken	Interface			
	decimals	External		-
	transfer	External	✓	-
	transferFrom	External	✓	-
SmarTradeContract	Implementation			
		Public	✓	-
	renewOwner	Public	✓	onlyOwner
	addPackages	Public	✓	onlyOwner
	addPackage	Public	✓	onlyOwner
	editPackage	Public	✓	onlyOwner
	getAllPackages	Public		-
	getStakingIdxForUser	Public		-
	getStakingReferIdxForUser	Public		-
	getActiveStakingsCount	Public		-
	getStakingsCount	Public		-
	getActiveStakingsAmount	Public		-
	stake	Public	✓	onlyEOA
	claim	Public	✓	onlyEOA

	claimEachStaking	Public	✓	onlyEOA onlyStaker
	unStake	Public	✓	onlyEOA onlyStaker
	calcRewards	Public		-
	depositToVault	Public	✓	onlyOwner
	isContract	Private		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SmarTradeContract
<Constructor>
initialize
setPrecision
setReferrerRate
onlyOwner
renewOwner
addPackages
addPackage
editPackage
getAllPackages
getStakingIdxForUser
getStakingReferIdxForUser
getActiveStakingsCount
getStakingsCount
getActiveStakingsAmount
stake
claim
claimEachStaking
calcRewards
unStake
depositToVault
onlyEOA
isContract
IToken (iface)
balanceOf
decimals
approve
transfer
transferFrom

Summary

Smartrade contract implements a staking 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