ACryptoS

SMART CONTRACT SECURITY ANALYSIS

Date of audit: 28.01.2021

Project Summary

Project Name	ACryptoS
Scope	A DeFi platform offering vaults with automated yield strategies and stablecoin trades
Platform	Binance Smart Chain, Solidity

Executive Summary

- Eight smart contracts were analyzed to check availability of code vulnerabilities associated with the process of funds staking:
  ACS, MasterChef, ACSI, MasterChefV2, StrategyACryptoS0V3, StrategyACryptoS0V4_ACSI, Controller and ACryptoSVault0.
- No significant security issues were revealed in the aforementioned contracts
- Currently, the project features a medium degree of centralization as the admin EOA has access to modification of the smart contracts

Smart Contract Ownership	Team Reward	Total Supply	Minting Function	Migration Function	Funds Lock Period	Contract Pause	Suspicious Functions
Controlled by the admin EOA	10% of user rewards	Variable	Available	Not available	None	Not available	Not found

External Smart Contract Audit:

- Not found

Manual Check Results

Ownership structure:

Smart contract	Owner	Description

ACS	Timelock6H
	MasterChef
		• Minters can mint tokens and add new minters;
		• Current minters are:
		  • Timelock6H (The admin is ACryptoS: Deployer, the current delay is 24h, the minimum delay is 6h);
		  • MasterChef;
		• The total supply is uncapped;
		• 8888.8... ACS tokens were preminted to ACryptoS: Deployer at the deployment stage.

MasterChef	Timelock6H	The following functions can be invoked by the owner:
		• **renounceOwnership** / **transferOwnership**;
		• **add** adds new LPs to the pool;
		• **set** updates the given pool's ACS allocation point;
		• **setSushiPerBlock** updates the reward rate, the current one is 0,08... ACS;
		• **setWithdrawalFee** sets a fee that is applied each time users withdraw their funds (exiting staking) or make a deposit (stake funds). The current fee can amount up to 7 ACSs and is paid out of pending user rewards. It cannot exceed withdrawal amounts and is not applicable to emergency exits;
		• **setAcsACS** sets the address receiving two types of fees: the 10% and 33.33% withdrawal fees. The current address is StrategyACryptoS0V3;
		• **setDevReward** updates the dev reward rate applied to each pool update (by each fund withdrawal). The current rate is 10%;
		• **setAcsACSReward** updates the acsACS reward applied to each pool updats (by each fund withdrawal). The current rate is 33.33%;
		• **dev** updates the dev address. The current one is ACryptoS: Deployer;
		• The total fees:
		  • 33.33% of rewards paid in ACS is transferred to StrategyACryptoS0V3;
		  • 10% of rewards paid in ACS is transferred to ACryptoS: Deployer;
		  • up to 7 ACSs paid from pending rewards to StrategyACryptoS0V3 by exiting the staking or depositing.

ACSI	Timelock6H
	MasterChefV2
		• Minters can mint tokens and add new minters;
		• The current minters are:
		  • Timelock6H (The admin is ACryptoS: Deployer, the current delay is 24h, the minimum delay is 6h);
		  • MasterChefV2;
		• The total supply is uncapped;
		• 8888.8... ACSIs where preminted to ACryptoS: Deployer at the deployment stage.

MasterChefV2		The following functions can be invoked by the owner: • **renounceOwnership** / **transferOwnership** • **add** adds new LPs to the pool; • **set** updates the given pool's ACS allocation point; • **setSushiPerBlock** updates the reward rate, the current one is 0,08... ACSI; • **setWithdrawalFee** sets a fee applied each time when users withdraw their funds (exit staking) or make deposits (stake funds). The current fee can amount up to 10 ACSIs paid out of pending user rewards. It cannot exceed withdrawal amounts and is not applicable to emergency exits; • **setAcsACS** sets the address receiving two types of fees: the 10% and 33.33% withdrawal fees. The current address is StrategyACryptoS0V4_ACSI; • **setDevReward** updates the dev reward rate applied to each pool update (by each fund withdrawal). The current rate is 10%;

		• **setAcsACSReward** updates the acsACS reward applied to each pool update (fund withdrawal). The current rate is 33.33%; • **dev** updates the dev address. The current address is ACryptoS: Deployer; The total fees: • 33.33% of rewards paid in ACSI is transferred to StrategyACryptoS0V4_ACSI; • 10% of rewards paid in ACSI s transferred to ACryptoS: Deployer; • up to 10 ACSIs paid out of pending rewards to StrategyACryptoS0V4_ACSI by exiting the staking or depositing.
StrategyACryptoS0V3	Timelock6H	The Timelock6H governance is ACryptoS: Deployer. The following functions can be invoked by the owner: • **setGovernance** sets new governance; • **setController** sets new controller; • **addSsToWithdraw** is used for strategy maintaining ; • **deleteSsToWithdraw** is used for strategy maintaining; • **addPairToLiquidate** is used for strategy maintaining; • **deletePairsToLiquidate** is used for strategy maintaining; • **addTokenToSwap0** / **addTokenToSwap1** is used for strategy maintaining; • **deleteTokensToSwap0** / **deleteTokensToSwap1** is used for strategy maintaining; • Controller can withdraw any asset, but he can't transfer ACS to itself; • According to the contract, 10% of ACS stay in strategy on withdrawal by user; • 0.3% of gains is transferred to harvesters (users, bots, etc.).
StrategyACryptoS0V4_ACSI	Timelock6H	The Timelock6H governance is ACryptoS: Deployer. The following functions can be invoked by the owner: • **setGovernance** sets new governance; • **setController** sets new controller; • **addSsToWithdraw** is used for strategy maintaining; • **deleteSsToWithdraw** is used for strategy maintaining; • **addPairToLiquidate** is used for strategy maintaining; • **deletePairsToLiquidate** is used for strategy maintaining; • **addTokenToSwap0** / **addTokenToSwap1** is used for strategy maintaining; • **deleteTokensToSwap0** / **deleteTokensToSwap1** is used for strategy maintaining; • Controller can withdraw any asset, but he can't transfer ACS to itself; • According to the contract, 10% of ACSI is kept in the strategy until withdrawn by the user; • 0.3% of gains is transferred to harvesters (users, bots, etc.).
Controller	Timelock6H	The Timelock6H governance is ACryptoS: Deployer. The following functions can be invoked by the owner: • **setRewards**; • **setStrategist**; • **setSplit**; • **setOneSplit**; • **setGovernance**; • **setVault** can be used to set vault address for a specific token; • **setConverter**; • **setStrategy**; • **approveStrategy** approves a previously settled strategy; • **revokeStrategy** disapproves a previously settled strategy; • **withdrawAll** withdraws all tokens of a strategy back to the relied vault; • **inCaseTokensGetStuck** withdraws all tokens from Controller to msg.sender; • **inCaseStrategyTokenGetStuck** withdraws all tokens from a strategy to the relied vault; • **yearn** allows to withdraw non-core strategy tokens; • **withdraw** withdraws tokens to vaults associated with these tokens.

ACryptoSVault0	Timelock6H	The Timelock6H governance is ACryptoS: Deployer. The following functions can be invoked by the owner: • **setMin** sets the denominator into the formula that defines the amount that can be borrowed; • **setGovernance**; • **setController**; • Controller can call harvesting - withdrawing non-core vault tokens to itself.

Total supply:
- Variable as long as new tokens are minted when users gain rewards

Minting function:
- Available

  The function can be executed with a 48-hour delay

Migration function:
- Not available

Team reward:
- 10% of user rewards

Funds lock period:
- None

Possibility to pause the Smart Contracts:
- Not available

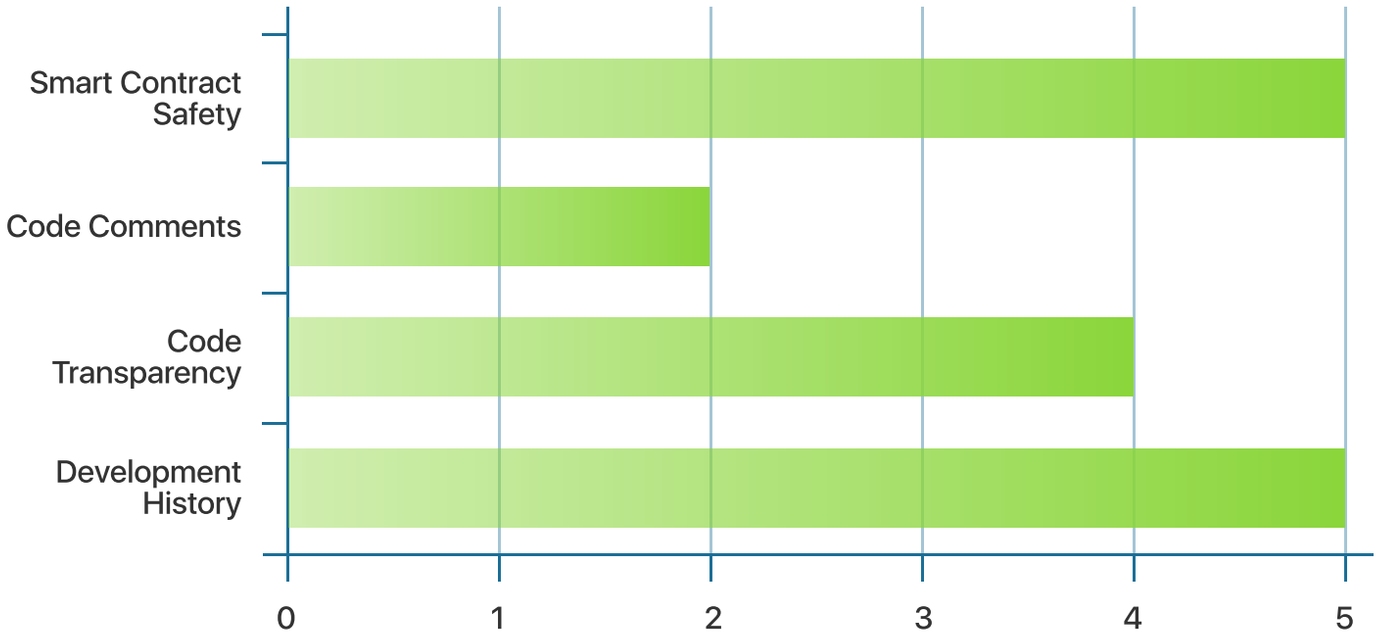
Suspicious functions:
- Not found

The risk of a quick token dump initiated by the team:
- **2** / 10

Tornado cash connections::
- Not found

Risk Level LOW

Smart Contracts



Conclusion

ACryptoS is a yield aggregator, which runs on Binance Smart Chain. The protocol is divided into Vaults (automated yield strategies), StableSwap (automated market maker) and Farms (a liquidity program where the ACS and ACSI tokens are distributed to liquidity providers).

The project has active governance realized through a voting mechanism presented on SnapShot. Holders of the ACS token are eligible to vote for new community proposals or support already existing terms. However, the admin EOA has direct access to the project smart contracts with possibility to call the underlying functions and change parameters of the contracts with a 48-hour timelock delay. This aspect can be seen in the following lines of code:

File 1 of 2: ACryptoSVault0.sol

```
412     }
413
414     function setController(address _controller) public {
415         require(msg.sender == governance, "!governance");
416         controller = _controller;
417     }
418
419     // Custom logic in here for how much the vault allows to be borrowed
```

Contract Source Code (Solidity)

```
1122
1123
1124    function setSushiPerBlock(uint256 _sushiPerBlock) external onlyOwner {
1125        sushiPerBlock = _sushiPerBlock;
1126    }
1127    function setAcsACS(address _acsACS) external onlyOwner {
1128        acsACS = _acsACS;
1129    }
1130    function setVault(address _vault) external onlyOwner {
1131        vault = _vault;
1132    }
1133    function setDisableHarvest(bool _disableHarvest) external onlyOwner {
1134        disableHarvest = _disableHarvest;
1135    }
1136    function setWithdrawalFee(uint256 _withdrawalFee) external onlyOwner {
1137        withdrawalFee = _withdrawalFee;
1138    }
1139    function setDevReward(uint256 _devReward) external onlyOwner {
1140        devReward = _devReward;
1141    }
1142    function setAcsACSReward(uint256 _acsACSReward) external onlyOwner {
1143        acsACSReward = _acsACSReward;
1144    }
1145
```

Before DefiYield started working on this audit, the timelock delay amounted to 24 hours only. Our team contacted developers of ACryptoS and explained the selected timelock had been insufficient for the user security. We are glad that ACryptoS took our suggestions into account having increased the timelock delay.

Fortunately, users can easily track any code changes using a timelock monitor and a telegram bot:
https://unrekt.net/acryptos/timelock.html
https://t.me/acryptos9/26174

Total supply of the project tokens is variable since new tokens are minted when users gain rewards (a Sushi-like system).

Contract Source Code (Solidity)

```
998     }
999
1000    // Update reward variables of the given pool to be up-to-date.
1001    function updatePool(uint256 _pid) public {
1002        PoolInfo storage pool = poolInfo[_pid];
1003        if (block.number <= pool.lastRewardBlock) {
1004            return;
1005        }
1006        uint256 lpSupply = pool.lpToken.balanceOf(address(this));
1007        if (lpSupply == 0) {
1008            pool.lastRewardBlock = block.number;
1009            return;
1010        }
1011        uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1012        uint256 sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
1013        sushi.mint(devaddr, sushiReward.mul(devReward).div(REWARD_DENOMINATOR));
1014        sushi.mint(acsACS, sushiReward.mul(acsACSReward).div(REWARD_DENOMINATOR));
1015        sushi.mint(address(this), sushiReward);
1016        pool.accSushiPerShare = pool.accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply));
1017        pool.lastRewardBlock = block.number;
1018    }
1019
1020    // Deposit LP tokens to MasterChef for SUSHI allocation.
```

The team reward amounts to 10% of the user rewards.

Contract Source Code (Solidity)

```
1009            return;
1010        }
1011        uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1012        uint256 sushiReward = multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
1013        sushi.mint(devaddr, sushiReward.mul(devReward).div(REWARD_DENOMINATOR));
1014        sushi.mint(acsACS, sushiReward.mul(acsACSReward).div(REWARD_DENOMINATOR));
1015        sushi.mint(address(this), sushiReward);
1016        pool.accSushiPerShare = pool.accSushiPerShare.add(sushiReward.mul(1e12).div(lpSupply));
1017        pool.lastRewardBlock = block.number;
1018    }
1019
```

No suspicious functions were revealed. The code of the analysed smart contracts is transparent. According to the aforementioned facts, the risk level can be estimated as low.

- This analysis is not a financial advice
- Conduct your own research before investing
- Track updates of yield farming platforms