Security Audit Report

ACoconut BTC (acBTC)

Phase Two: Migration and Swap Application

SECBIT

Oct 29, 2020

1. Introduction

ACoconut BTC (acBTC) is a synthetic BTC ERC20 Token on Ethereum. acBTC integrates native BTC and ERC-20 BTC, with the swap, lending, and yield generating applications into one highly secure, efficient, and usable standard. SECBIT Labs conducted an audit from Sep 10th to Oct 22nd, 2020, including an analysis of Smart Contracts in 3 areas: **code bugs**, **logic flaws**, and **risk assessment**. The audit results show that ACoconut BTC (acBTC) Phase Two has no critical security risks. The SECBIT team has some tips on logical implementation, potential risks, and code revising (see part 4 for details).

Type	Description	Level	Status
Logical Implementation	4.3.1 In the ACoconutSwap contract, the method of calculating the accuracy of `dy` is inconsistent.	Info	Fixed
Logical Implementation	4.3.2 The method of burning acBTC of users in ACoconutSwap contract is in doubt.	Low	Fixed
Logical Implementation	4.3.3 The approveToken in the SwapApplication::mintToken function may accidentally clear the approved allowance.	Low	Fixed
Code Revise	4.3.4 There is room for gas optimization in the StakingApplication contract.	Info	Fixed
Code Revise	4.3.5 There is room for gas optimization in the SwapApplication contract.	Info	Fixed
Code Revise	4.3.6 Consider increasing the number of solidity compilation and optimization runs.	Info	Fixed

2. Project Information

This part describes the necessary information and code structure.

2.1 Basic information

The basic information about acBTC is shown below:

- Project website
    - https://acbtc.fi
- Design details of the protocol
    - https://docs.acbtc.fi
- Smart contract code
    - https://github.com/nutsfinance/acBTC, commit e9090951e8a7a016563995c81197c3254817dda3

2.2 Contract List

The following content shows the main contracts included in acBTC project:

Contract	Description
Account.sol	Trading Account Contract
AccountFactory.sol	Trading account generation contract
ACoconut.sol	AC Token Contract
ACoconutBTC.sol	acBTC Token contract
ACoconutSwap.sol	Implementation of acSwap
ACoconutSwapProxy.sol	The proxy contract for the deployment of acSwap
ACoconutVault.sol	Implementation of acVault
CurveRenCrvMigrator.sol	RenCRV migration contract
StrategyACoconutBTC.sol	acBTC earning strategy contract
StakingApplication.sol	Implementation of Staking Application
AdminUpgradeabilityProxy.sol	Upgradeable proxy contract with authorization
BaseUpgradeabilityProxy.sol	Upgradeable proxy contract
Initializable.sol	Initialization control contract
Proxy.sol	Proxy contract
Controller.sol	Vault controller contract
RewardedVault.sol	Rewarded vault contract
StrategyCurveRenBTC.sol	RenCRV earning strategy contract
Vault.sol	Vault contract
ACoconutMaker	Contract that collects transaction fees from ACoconutSwap
SwapApplication	Implementation of Swap Application

3. Code Analysis

This part describes code assessment details, including two items: "role classification" and "functional analysis".

3.1 Role Classification

There are several vital roles in acBTC Phase Two, namely Governance Account, Minter, Wallet Account, and Common Account.

- Governance Account
    - Description Contract governor
    - Authority
        - Set the minter of AC Token or acBTC Token
        - Set the basic parameters of the contract
    - Method of Authorization The creator of the smart contract, or authorized by the transferring of governance account
- Minter
    - Description The accounts minting AC Token or acBTC Token
    - Authority
        - Mint
    - Method of Authorization Authorized by governance account
- Wallet Account
    - Description The account used to interact with the acBTC contract
    - Authority
        - Hierarchical authority management (owner, admin, operator)
        - Owner account can grant or revoke admin accounts
        - Admin account can grant or revoke operator accounts
        - Operator account can transfer ETH and ERC20 Token in the wallet account
        - Operator account can approve ERC20 Token in the wallet account
        - Operator account can transfer the ERC20 Token of the owner account when approved by the owner account
        - Operator account can invoke delegate calls
    - Method of Authorization
        - Owner account is the creator of the contract or authorized by the

        transferring of the owner account
        - Admin account is authorized by the owner account
        - Operator account is authorized by the admin account
- Common Account
    - Description The accounts holding AC Token or acBTC Token
    - Authority
        - Transfer tokens in its account
        - Approve other accounts to transfer
        - AC Token holders can participate in the governance votes
    - Method of Authorization AC Token or acBTC holders

3.2 Functional Analysis

acBTC is a synthetic BTC ERC20 Token contract. We can divide the critical functions of the acBTC contract into several parts:

- acBTC

acBTC is a synthetic ERC20 BTC token backed by a basket of ERC20 BTC tokens. It's built on top of Curve's StableSwap algorithm with integrated saving and swap applications.

- acVault

acVault is a renCrv vault that earns yields in renCrv before migration and becomes an acBTC vault that makes yields in acBTC after migration.

- acSwap

acSwap is a decentralized exchange for ERC20 BTC tokens. It manages a basket of ERC20 BTC tokens, including WBTC and renBTC, and bootstraps the value of acBTC.

acSwap provides the following functions:

Minting acBTC

- Users can mint new acBTC by depositing any amount of ERC20 BTC tokens, similar to adding liquidity into Curve.fi's swap pool.

Redeeming acBTC

- Users can redeem their acBTC to reclaim underlying ERC20 BTC tokens. acSwap defines three ways to redeem acBTC：
    - Users define the number of acBTC to redeem and the minimum number of underlying tokens to receive.
    - Users define the number of acBTC to redeem and want to redeem to a single

token.
    - Users define the number of underlying tokens to receive as well as the maximum acBTC to burn.

Swap

- Similar to Curve.fi, acSwap allows swap between any supported pair of underlying ERC20 BTC tokens. acSwap ensures that the value D is unchanged before and after the swap.

4. Audit Detail

This part describes the process and detailed results of the audit and demonstrates the problems and potential risks.

4.1 Audit Process

The audit strictly followed the audit specification of SECBIT Labs. We analyzed the project from code bug, logical implementation, and potential risks. The process consists of four steps:

- Fully analysis of code line by line.
- Evaluation of vulnerabilities and potential risks revealed in the source code.
- Communication on assessment and confirmation.
- Audit report writing.

4.2 Audit Result

After scanning with adelaide, sf-checker, and badmsg.sender (internal version) developed by SECBIT Labs and open source tools including Mythril, Slither, SmartCheck, and Securify, the auditing team performed a manual assessment. The team inspected the contract line by line, and the result could be categorized into twenty-one types:

Number	Classification	Result
1	Normal functioning of features defined by the contract	✓
2	No obvious bug (e.g., overflow, underflow)	✓

3	Pass Solidity compiler check with no potential error	✓
4	Pass common tools check with no obvious vulnerability	✓
5	No obvious gas-consuming operation	✓
6	Meet with ERC20	✓
7	No risk in low-level call (call, delegatecall, callcode) and in-line assembly	✓
8	No deprecated or outdated usage	✓
9	Explicit implementation, visibility, variable type, and Solidity version number	✓
10	No redundant code	✓
11	No potential risk manipulated by timestamp and network environment	✓
12	Explicit business logic	✓
13	Implementation consistent with annotation and other info	✓
14	No hidden code about any logic that is not mentioned in design	✓
15	No ambiguous logic	✓
16	No risk threatening the developing team	✓
17	No risk threatening exchanges, wallets, and DApps	✓
18	No risk threatening token holders	✓
19	No privilege on managing others' balances	✓
20	No unnecessary minting method	✓
21	Correct managing hierarchy	✓

4.3 Issues

4.3.1 In the ACoconutSwap contract, the method of calculating the accuracy of `dy` is inconsistent.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Info	Calculation accuracy	Fixed

Description

The detail of calculating the accuracy of `dy` `getRedeemSingleAmount()` and `redeemSingle()` functions is inconsistent with that in `getSwapAmount()`.

```
// ACoconutSwap.sol
uint256 dy = _balances[_i].sub(y).div(precisions[_i]); // L410, L443
uint256 dy = _balances[_j].sub(y).sub(1).div(precisions[_j]); // L266,
L297
```

Suggestion

It is recommended to follow the method with `sub(1)` in Curve.

Status

It has been fixed according to the suggestion in 991662d.

4.3.2 The method of burning acBTC of users in ACoconutSwap contract is in doubt.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Low	Permission control	Fixed

Description

In the ACoconutSwap contract, when the acBTC of users is burned, the minter account can directly burn any user's balance, which may cause user concerns.

```
// ACoconutSwap.sol
IERC20MintableBurnable(poolToken).burn(msg.sender, _amount); // L384,
L452, L523
```

Suggestion

It is recommended to first `approve()` and then `burnFrom()` in ACoconutSwap contract to execute the burning operation.

Status

It has been fixed according to the suggestion in 9b4e19.

4.3.3 The approveToken in the SwapApplication::mintToken function may accidentally clear the approved allowance.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Low	Unexpected approval	Fixed

Description

There is a cyclic approve operation in the `mintToken()` function to set the user's allowance. When the user uses a single asset to perform mint operations, the corresponding index values of other assets in the `_amounts` array are set to 0, which may accidentally set the corresponding allowances to 0.

```
// SwapApplication.sol
function mintToken(address _account, uint256[] memory _amounts,
uint256 _minMintAmount) public validAccount(_account) {
    Account account = Account(payable(_account));
    // We don't perform input validations here as they are done in
ACoconutSwap.
    for (uint256 i = 0; i < _amounts.length; i++) {
        account.approveToken(swap.tokens(i), address(swap),
_amounts[i]);
    }

        bytes memory methodData =
abi.encodeWithSignature("mint(uint256[],uint256)", _amounts,
_minMintAmount);
        account.invoke(address(swap), 0, methodData);
    }
```

Suggestion

It is recommended to add a judgment condition in the loop, as follows:

```
for (uint256 i = 0; i < _amounts.length; i++) {
  if (_amounts[i]) continue;
  account.approveToken(swap.tokens(i), address(swap), _amounts[i]);
}
```

Status

It has been fixed according to the suggestion in 7490c0.

4.3.4 There is room for gas optimization in the StakingApplication contract.

Risk Type	Risk Level	Impact	Status
Code Revise	Info	gas optimization	Fixed

Description

In the StakingApplication contract, some interfaces that will only be called externally can be declared as `external` to save gas.

- initialize
- setGovernance
- setController
- stake
- unstake
- exit
- getVaultBalance
- getStakeBalance
- getUnclaimedReward
- getClaimedReward

Suggestion

It is recommended to declare the interface in the list as `external`.

Status

It has been fixed according to the suggestion in 98fb1b.

4.3.5 There is room for gas optimization in the SwapApplication contract.

Risk Type	Risk Level	Impact	Status
Code Revise	Info	gas optimization	Fixed

Description

In the SwapApplication contract, some interfaces that will only be called externally can be declared as `external` to save gas.

- initialize
- setGovernance
- setSwap
- mintToken
- swapToken
- redeemProportion
- redeemSingle
- redeemMulti

Suggestion

It is recommended to declare the interface in the list as `external`.

Status

It has been fixed according to the suggestion in d73d9f.

4.3.6 Consider increasing the number of solidity compilation and optimization runs.

Risk Type	Risk Level	Impact	Status
Code Revise	Info	Compilation optimization	Fixed

Description

The current setting is：

```
// truffle-config.js
optimizer: {
  enabled: true,
  runs: 250
}
```

Suggestion

It is recommended to increase the value of `runs`. A single interface's gas consumption may drop by thousands, which is more meaningful for contracts with high frequency.

Status

It has been fixed according to the suggestion in 749ab20.

4.4 Risks

4.4.1 In ACoconutSwap contract, the `collectFee()` function may affect the asset ratio of swap pool in extreme cases.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Info	Asset ratio of swap pool	Discussed

Description

The `collectFee()` function is used to collect transaction fees and refreshes the `balances` array. The admin account manually invokes this interface. If the accumulated balance difference is quite large, it may impact the swap pool's asset ratio. No security risk is found here, and the admin should often call `collectFee()`.

```
// ACoconutSwap.sol
function collectFee() external returns (uint256) {
    require(admins[msg.sender], "not admin");
    uint256[] memory _balances = balances;
    uint256 A = getA();
    uint256 oldD = totalSupply;

    for (uint256 i = 0; i < _balances.length; i++) {
        _balances[i] =
IERC20(tokens[i]).balanceOf(address(this)).mul(precisions[i]);
    }
    uint256 newD = _getD(_balances, A);
    uint256 feeAmount = newD.sub(oldD);
    if (feeAmount == 0) return 0;

    balances = _balances;
    totalSupply = newD;
    address _feeRecipient = feeRecipient;
    IERC20MintableBurnable(poolToken).mint(_feeRecipient, feeAmount);

        emit FeeCollected(_feeRecipient, feeAmount);

        return feeAmount;
    }
```

Suggestion

None.

Status

Discussed.

4.4.2 StakingApplication and SwapApplication should not save important state and assets in the future.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Info	Protocol Mechanism	Discussed

Description

StakingApplication and SwapApplication interact with any user-selected contract and can be upgraded. These two contracts are only used as helper contracts for the interaction between the account contract and the application contract. At present, no security issues have been found. Since the user could freely set the account address, the address may belong to an unknown contract and have potential risks.

Suggestion

It is recommended not to store important status or assets in StakingApplication and SwapApplication contracts or conduct stricter authentication and inspections when upgrading in the future.

Status

Discussed.

4.4.3 The modification of the value of A and the addition or deletion of the underlying BTC token need to be cautious.

Risk Type	Risk Level	Impact	Status
Logical Implementation	Info	Protocol Mechanism	Discussed

Description

Modifying A's value, adding or deleting the underlying BTC token, involves changes in the Token Bonding Curve model's details and must be modified under the premise of comprehensive research and testing. Currently, the ACoconutSwap contract does not support modification, and future updates must be particularly cautious.

Suggestion

None.

Status

Discussed.

5. Conclusion

After auditing and analyzing the contract code of it, SECBIT Labs found some issues to optimize and proposed corresponding suggestions, which have been shown above. The acBTC developers have them all fixed in the code's latest version for the issues found in this report. SECBIT Labs holds that the acBTC project has high code quality, detailed documentation, and complete test cases. acBTC Phase Two fully implements acBTC migration and acSwap function. The acSwap provides mint, redeem, and swap functions, opening up BTC Token circulation and promoting the development of DeFi applications.

Disclaimer

APPENDIX

Vulnerability/Risk Level Classification

Level	Description
High	Severely damage the contract's integrity and allow attackers to steal ethers and tokens or lock ethers inside the contract.
Medium	Damage contract's security under given conditions and cause impairment of benefit for stakeholders.
Low	Cause no actual impairment to contract.
Info	Relevant to practice or rationality could possibly bring risks.

http://www.secbit.io

audit@secbit.io

@secbit_io