智能合约安全审计报告

ACoconut BTC（acBTC）

Phase Two: Migration and Swap Application

SECBIT

Oct 29, 2020

安比（SECBIT）实验室致力于解决区块链全生态的安全问题，提供区块链全生态的安全服务。作为中国信息通信研究院区块链安全技术组成员，参与编写区块链安全白皮书和参与制定区块链安全审计规范。

安比（SECBIT）实验室智能合约审计从合约的技术实现、业务逻辑、接口规范、Gas 优化、发行风险等维度，由两组专业审计人员分别独立进行安全审计，审计过程借助于安比实验室研发的一系列形式化验证、语义分析等工具进行扫描检测，力求尽可能全面的解决所有安全问题。

1. 综述

ACoconut BTC（acBTC）是基于以太坊的聚合 BTC ERC20 合约。acBTC 协议支持原生 BTC 以及 BTC ERC20 代币的交换、贷款和其他产生收益的应用，并集成到一个高度安全，高效和可用的标准中。安比（SECBIT）实验室于 2020 年 10 月 10 日至 10 月 22 日对 acBTC Phase Two 合约进行安全审计，审计过程从**代码漏洞**，**逻辑漏洞**和**发行风险评估**三个维度对代码进行分析。审计结果表明，acBTC Phase Two 并未包含致命的安全漏洞，安比（SECBIT）实验室给出了如下几点逻辑实现存疑和代码优化建议项（详见第4章节）。

风险类型	描述	风险级别	状态
代码实现	4.3.1 ACoconutSwap 合约中，计算 dy 处理精度的方法前后不一致。	提示	已修复
代码实现	4.3.2 ACoconutSwap 合约中销毁用户 acBTC 的方式存疑。	低	已修复
代码实现	4.3.3 SwapApplication::mintToken 函数中 approveToken 可能会意外清空已授权的 allowance。	低	已修复
代码优化	4.3.4 StakingApplication 合约存在 gas 优化空间。	提示	已修复
代码优化	4.3.5 SwapApplication 合约存在 gas 优化空间。	提示	已修复
代码优化	4.3.6 可考虑提高 solidity 编译优化 runs 次数。	提示	已修复

2. 项目信息

该部分描述了项目的基本信息和代码组成。

2.1 基本信息

以下展示了 acBTC 的基本信息：

- **项目网站**
  - https://acbtc.fi
- **项目设计文档**
  - https://docs.acbtc.fi/
- **合约代码**
  - https://github.com/nutsfinance/acBTC, commit e9090951e8a7a016563995c81197c3254817dda3

2.2 合约列表

以下展示了 acBTC 项目包含的主要合约列表：

合约名称	描述
Account	交易账户合约
AccountFactory	交易账户生成合约
ACoconut	AC Token合约
ACoconutBTC	acBTC Token合约
ACoconutSwap	acSwap功能实现
ACoconutSwapProxy	acSwap部署使用的proxy合约
ACoconutVault	acVault功能实现
CurveRenCrvMigrator	renCrv转移合约
StrategyACoconutBTC	acBTC收益策略合约
StakingApplication	权益应用功能实现
AdminUpgradeabilityProxy	授权功能的proxy升级合约
BaseUpgradeabilityProxy	proxy升级合约
Initializable	初始化控制合约
Proxy	Proxy合约
Controller	资金管理合约
RewardedVault	有奖励的资金收益管理合约
StrategyCurveRenBTC	renCrv收益策略合约
Vault	资金收益管理合约
ACoconutMaker	acSwap交易费用收集
SwapApplication	acSwap应用功能实现

3. 代码分析

该部分描述了项目代码的详细内容分析，从「角色分类」和「功能分析」这两部分来进行说明。

3.1 角色分类

acBTC Phase Two 中主要涉及以下几种关键角色，分别是 Governance Account（治理账户）、Minter（铸币者）、Wallet Account（钱包账户）、Common Account（普通账户）。

- 治理账户 (Governance Account)
  - 描述

    合约的治理者
  - 功能权限
    - 设定 AC Token 或 acBTC 的铸币者
    - 设定合约的基础参数
  - 授权方式

    合约的创建者，或者由治理账户转让授权
- 铸币者 (Minter)
  - 描述

    对 AC Token 或 acBTC Token 进行铸币操作
  - 功能权限

    铸币
  - 授权方式

    由治理账户授权
- 钱包账户 (Wallet Account)
  - 描述

    用户与 acBTC 合约交互使用的账户
  - 功能权限
    - 分级权限管理（owner, admin, operator）
    - owner 账户授权或取消 admin 账户权限
    - admin 账户授权或取消 operator 账户权限
    - operator 账户对钱包账户中的 ETH 和 ERC20 Token 进行转账操作
    - operator 账户对钱包账户中的 ERC20 Token 进行授权操作
    - operator 账户在owner授权后对其账户 ERC20 Token 进行转账操作
    - operator 账户可发起远程调用

    - 授权方式
        - owner 为合约的创建者，或者由 owner 转让授权
        - admin 由 owner 授权
        - operator 由 admin 授权
- 普通账户（Common Account）
    - 描述

      持有 AC Token 或 acBTC Token 的账户
    - 功能权限
        - 对账户上的 AC Token 或 acBTC Token 进行转账
        - 授权其他账户转账
        - AC Token 持有者参与投票治理
    - 授权方式

      AC Token 或 acBTC Token 的持有者

3.2 功能分析

acBTC 是基于以太坊的聚合 ERC20 BTC Token 合约，合约实现关键功能分为以下几项：

- **acBTC**

  acBTC 是 聚合 BTC ERC20 Token 的合约，基于 Curve 的 StableSwap 算法集成了 BTC ERC20 Token 存储与交换应用。
- **acVault**

  acVault 是 renCrv 的资金库，抵押 renCrv 的用户可在 renCrv 迁移前，从合约中获得收益。renCrv 迁移后成为 acBTC 资金库，用户可在 acBTC 中获得收益。
- **acSwap**

  acSwap 是 ERC20 BTC Token 的去中心化交易所，它管理聚合的 ERC20 BTC Token，包括 WBTC 和 renBTC，并从中引导 acBTC 的价值。

  acSwap提供以下功能：

  铸造 acBTC
    - 用户可以通过存入任意数量的ERC20 BTC Token 来铸造新的acBTC，这类似于在Curve.fi的交换池中增加流动性。

  赎出 acBTC
    - 用户可以赎出其 acBTC 来换回 ERC20 BTC Token，有以下三种方式：
        - 给定赎出的 acBTC 的数量，换回组合Token比例的 ERC20 BTC Token
        - 给定赎出的 acBTC 的数量，换回指定的 ERC20 BTC Token
        - 给定 ERC20 BTC Token 数量，赎出对应数量的 acBTC

  swap

- 与 Curve.fi 类似，acSwap 允许在其支持的任何一对 ERC20 BTC Token 之间交换，acSwap 确保在交换前后 D 值保持不变。

4. 审计详情

该部分描述合约审计流程和详细结果，并对发现的问题（代码漏洞，代码规范和逻辑漏洞），合约发行的风险点和附加提示项进行详细的说明。

4.1 审计过程

本次审计工作，严格按照安比（SECBIT）实验室审计流程规范执行，从代码漏洞，逻辑问题以及合约发行风险三个维度进行全面分析。审计流程大致分为四个步骤:

- 各审计小组对代码进行逐行分析，根据审计内容要求进行审计
- 各审计小组对漏洞和风险进行评估
- 审计小组之间交换审计结果，并对审计结果进行逐一审查和确认
- 审计小组配合审计负责人生成审计报告

4.2 审计结果

本次审计首先经过安比（SECBIT）实验室推出的分析工具 adelaide、sf-checker 和 badmsg.sender（内部版本）扫描，再利用开源安全分析工具 Mythril、Slither、SmartCheck 以及 Securify 检查，检查结果由审计小组成员详细确认。审计小组成员对合约源码和电路代码进行逐行检查、评估，汇总审计结果。审计内容总结为如下 21 大项。

编号	分类	结果
1	合约各功能能够正常执行	通过
2	合约代码不存在明显的漏洞（如整数溢出）	通过
3	能够通过编译器的编译并且编译器没有任何警告输出	通过
4	合约代码能够通过常见检测工具检测，并无明显漏洞	通过
5	不存在明显的 Gas 损耗	通过
6	符合 EIP20 标准规范	通过

7	底层调用（call, delegatecall, callcode）或内联汇编的操作不存在安全隐患	通过
8	代码中不包含已过期或被废弃的用法	通过
9	代码实现清晰明确，函数可见性定义明确，变量数据类型定义明确，合约版本号明确	通过
10	不存在冗余代码	通过
11	不存在受时间和外部网络环境影响的隐患	通过
12	业务逻辑实现清晰明确	通过
13	代码实现逻辑与注释，项目白皮书等资料保持一致	通过
14	代码不存在设计意图中未提及的逻辑	通过
15	业务逻辑实现不存在疑义	通过
16	不存在危及项目方利益的明确风险	通过
17	不存在危及相关机构如交易所，钱包，DAPP 方利益的明确风险	通过
18	不存在危及普通持币用户利益的明确风险	通过
19	不存在修改他人账户余额的特权	通过
20	不存在非必要的铸币权限	通过
21	多管理角色下，管理权限划分明确，各权限优先级划分明确	通过

4.3 问题列表

4.3.1 ACoconutSwap 合约中，计算 dy 处理精度的方法前后不一致。

风险类型	风险级别	影响点	状态
代码实现	提示	计算精度	已修复

问题描述

`getRedeemSingleAmount()` 和 `redeemSingle()` 函数计算 `dy` 处理精度细节上与 `getSwapAmount()` 中不一致。

```
// 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
```

修改建议

建议按照统一按照 curve 中 `sub(1)` 的方法处理。

状态

已在 991662d 中按照修改建议修复。

4.3.2 ACoconutSwap 合约中销毁用户 acBTC 的方式存疑。

风险类型	风险级别	影响点	状态
代码实现	低	权限管理	已修复

问题描述

ACoconutSwap 合约中，销毁用户 acBTC 时，采用 minter 权限账户直接 burn 任意用户 balance 的方式，可能引起用户顾虑。

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

修改建议

建议由用户先 `approve()` 再在 ACoconutSwap 合约中 `burnFrom()` 执行销毁。

状态

已在 9b4e19 中按照修改建议修复。

4.3.3 SwapApplication::mintToken 函数中 approveToken 可能会意外清空已授权的 allowance。

风险类型	风险级别	影响点	状态
代码实现	低	意外授权	已修复

问题描述

`mintToken()` 函数中存在一处循环 approve 操作，设置用户的 allowance 。用户使用单个资产进行 mint 操作时，`_amounts` 数组中其他资产相应 index 的值被置为 0，从而可能会意外地将相应 allowance 均置为 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);
    }
```

修改建议

建议在循环中添加一个判断条件，如下：

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

状态

已在 7490c0 中按照修改建议修复。

4.3.4 StakingApplication 合约存在 gas 优化空间。

风险类型	风险级别	影响点	状态
代码优化	提示	gas优化	已修复

问题描述

StakingApplication 合约中，部分仅会被外部调用的接口可声明为 `external` 来节省 gas。

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

修改建议

建议将列表中的接口声明为 `external` 。

状态

已在 98fb1b 中按照修改建议修复。

4.3.5 SwapApplication 合约存在 gas 优化空间。

风险类型	风险级别	影响点	状态
代码优化	提示	gas优化	已修复

问题描述

SwapApplication 合约中，部分仅会被外部调用的接口可声明为 `external` 来节省 gas。

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

修改建议

建议将列表中的接口声明为 `external` 。

状态

已在 d73d9f 中按照修改建议修复。

4.3.6 可考虑提高 solidity 编译优化 runs 次数。

风险类型	风险级别	影响点	状态
代码优化	提示	编译优化	已修复

问题描述

目前设置为:

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

修改建议

建议提高 `runs` 数值，单个接口 gas 消耗可能会下降上千，对于高频使用的合约较有意义。

状态

已在 749ab20 中按照修改建议修复。

4.4 风险提示

4.4.1 ACoconutSwap 合约中，`collectFee()` 函数在极端情况下，可能会对资金池比例造成一定影响。

风险类型	风险级别	影响点	状态
代码实现	提示	资金池比例	已讨论

问题描述

`collectFee()` 函数用于提取交易费用，同时会刷新 `balances` 数组。该接口为管理员手动调用，如果累计的 balance 差别较大，则可能对资金池资产比例造成一定影响。未发现此处存在安全隐患，admin 经常调用 `collectFee()` 即可。

```
// 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;
    }
```

修改建议

无。

状态

已讨论。

4.4.2 StakingApplication 和 SwapApplication 未来不应保存重要状态和资产。

风险类型	风险级别	影响点	状态
代码实现	提示	协议机制	已讨论

问题描述

目前 StakingApplication 和 SwapApplication 与任意用户指定的合约交互，并且均可升级。目前这两个合约仅作为 account 合约与业务合约交互的 helper 合约，未发现安全问题。由于 account 地址由用户自由传入，属于未知合约，可能存在潜在风险。

修改建议

建议未来升级时，不在 StakingApplication 和 SwapApplication 合约中保存重要状态或资产，或进行更严格的鉴权和检查。

状态

已讨论。

4.4.3 SwapApplication中， A 值的修改以及底层 BTC token 的增加或修改需要谨慎。

风险类型	风险级别	影响点	状态
代码实现	提示	协议机制	已讨论

问题描述

修改 A 值，增删底层 BTC token，涉及到 Token Bonding Curve 模型的细节变更，必须在充分研究和测试的前提下进行修改。目前 ACoconutSwap 合约尚不支持修改，未来更新须特别谨慎。

修改建议

无。

状态

已讨论。

5. 结论

安比（SECBIT）实验室在对 ACoconut BTC（acBTC）合约进行分析后，发现部分可优化项，并提出了对应的修复及优化建议，上文均已给出具体的分析说明。对于本报告中提出的问题，acBTC 开发者均已在最新版代码中进行了修复。安比（SECBIT）实验室认为 acBTC 项目代码质量较高、文档详细、测试用例完整。acBTC Phase Two 完整实现了 acBTC Migration 和 acSwap 功能，acSwap 中提供了 mint, redeem, swap 功能，打通了 BTC Token 的流通，促进 DeFi 应用发展。

免责声明

SECBIT 智能合约安全审计从合约代码质量、合约逻辑设计和合约发行风险等方面对合约的正确性、安全性、可执行性进行审计，但不做任何和代码的适用性、商业模式和管理制度的适用性及其他与合约适用性相关的承诺。本报告为技术信息文件，不作为投资指导，也不为代币交易背书。

附录

漏洞风险级别介绍

风险级别	风险描述
高	可以严重损害合约完整性的缺陷，能够允许攻击者盗取以太币及Token，或者把以太币锁死在合约里等缺陷。
中	在一定限制条件下能够损害合约安全的缺陷，造成某些参与方利益损失的缺陷。
低	并未对合约安全造成实质损害的缺陷。
提示	不会带来直接的风险，但与合约安全实践或合约合理性建议有关的信息。

安比（SECBIT）实验室致力于参与共建共识、可信、有序的区块链经济体。

https://secbit.io

info@secbit.io

@secbit_io