智能合约安全审计报告

1. 概要.....1

2. 审计方法.....2

3. 项目背景.....3

3.1 项目介绍.....3

3.2 项目结构.....3

3.3 项目架构.....5

4. 代码概述.....5

4.1 主要合约地址.....5

4.2 主要合约函数权限分析.....6

4.3 代码审计详情.....7

4.3.1 中危漏洞.....7

4.3.2 低危漏洞.....7

4.3.3 增强建议.....7

5. 审计结果.....9

5.1 总结.....9

6. 声明.....10

1. 概要

慢雾安全团队于 2021 年 03 月 01 日，收到 DeFiBOX 团队对 lend 系统安全审计的申请，根据项目特点慢雾安全团队制定如下审计方案。

慢雾安全团队将采用“白盒为主，黑灰为辅”的策略，以最贴近真实攻击的方式，对项目进行安全审计。

慢雾科技 DeFi 项目测试方法：

黑盒测试	站在外部从攻击者角度进行安全测试。
灰盒测试	通过脚本工具对代码模块进行安全测试，观察内部运行状态，挖掘弱点。
白盒测试	基于项目的源代码，进行脆弱性分析和漏洞挖掘。

慢雾科技 DeFi 漏洞风险等级：

严重漏洞	严重漏洞会对项目的安全造成重大影响，强烈建议修复严重漏洞。
高危漏洞	高危漏洞会影响项目的正常运行，强烈建议修复高危漏洞。
中危漏洞	中危漏洞会影响项目的运行，建议修复中危漏洞。
低危漏洞	低危漏洞可能在特定场景中会影响项目的业务操作，建议项目方自行评估和考虑这些问题是否需要修复。
弱点	理论上存在安全隐患，但工程上极难复现。
增强建议	编码或架构存在更好的实践方法。

2. 审计方法

慢雾安全团队智能合约安全审计流程包含两个步骤:

- 使用开源或内部自动化分析的工具对合约代码中常见的安全漏洞进行扫描和测试。
- 人工审计代码的安全问题，通过人工分析合约代码，发现代码中潜在的安全问题。

如下是合约代码审计过程中我们会重点审查的漏洞列表:

（其他未知安全漏洞不包含在本次审计责任范围）

- 溢出审计
- 权限漏洞审计
- 权限过大审计
- 硬编码地址安全
- 显现编码安全
- 异常校验审计
- 类型安全审计
- 性能优化审计
- 设计逻辑审计
- 拒绝服务审计
- 回滚攻击审计
- 重放攻击审计
- 假通知审计
- 假错误通知审计
- 假币审计
- 随机数安全审计
- 粉尘攻击安全审计
- 微分叉安全审计
- 排挤攻击安全审计
- 重入攻击安全审计

3. 项目背景

3.1 项目介绍

Lend 是 DeFiBOX 推出的资产抵押借贷平台。

审计合约文件：

项目源代码

审计初始版本：

SHA256(lend-contracts-2021-02-25.zip)=

acf59eea54c0e47954abeb1342dcef8027148e33e0b95e071f91674745ad9750

审计最终版本：

SHA256(lend-2021-04-01.zip)=

9e8e77f3cb33764f3aa46c6c867ea50feed36bc0b345c1c85fbce8c0ff3fc8bd

SHA256(lend-contracts-2021-04-27.zip)=

2d533dff8e208fc90a3bf917d2116c89ce7046a64fb4058d3a30dca2acf02f84

SHA256(lend-2021-04-29.zip)=

3b521bc8a866451498fd5dcce2ae8a145b1e51ec0441f0eae914fd6c76e7b4bf

SHA256(lend-2021-05-26.zip)=

dde97ce95e80b699d143912a8434f3d40b5727a1ac0dc660c2d13866f7ccb43a

3.2 项目结构

.

```
├── CMakeLists.txt
├── btoken
│   ├── CMakeLists.txt
│   ├── README.txt
│   ├── include
│   │   ├── btoken.hpp
│   │   └── butils.hpp
│   ├── ricardian
│   │   └── btoken.contracts.md
│   └── src
│       └── btoken.cpp
├── common
│   └── include
│       ├── core.hpp
│       ├── defines.hpp
│       ├── safemath.hpp
│       ├── wadmath.hpp
│       └── wadraymath.hpp
├── lend
│   ├── CMakeLists.txt
│   ├── README.txt
│   ├── include
│   │   ├── lend.hpp
│   │   ├── structs.hpp
│   │   └── utils.hpp
│   ├── lend-cmd.txt
│   ├── ricardian
│   │   ├── lend.clauses.md
│   │   └── lend.contracts.md
│   └── src
│       └── lend.cpp
└── log
    ├── CMakeLists.txt
    ├── include
    │   └── log.hpp
    ├── ricardian
    │   └── log.contracts.md
    └── src
        └── log.cpp
```

3.3 项目架构

lend 项目主要包含 3 个合约，代币合约、借贷合约、日志合约。用户通过将代币转入借贷合约，获得代币凭证，同时可使用抵押品借出其它代币，在不卖出抵押品的同时提高资本的利用率。

4. 代码概述

4.1 主要合约地址

2021-04-01:

Contract Name	Code Hash(eosio.cdt v1.6.3)
btoken	9e10b8590f05dfe67c69272e41e9c719e04545de7b3ba4c99c2b2f9549ae5f51
lend	811dda9f2fb347a56216b788f287bfb146f15ea3a7dc479dcf0e849d89732f28
log	3c8126345418df2225f1deb70108876c825b50cedecf59c5acac760e6d6aba90

2021-04-27:

Contract Name	Code Hash(eosio.cdt v1.6.3)
btoken	436b792ca8a50415258542346dcdee4509c18cc5f507ecfef2eee1cb7b7ef7f4
lend	29e5f5e0731559a2bfaede1f7bb0dfaec119d785c21810b7ea2ecc3ddd8782b3
log	3c8126345418df2225f1deb70108876c825b50cedecf59c5acac760e6d6aba90

2021-04-29:

Contract Name	Code Hash(eosio.cdt v1.6.3)
btoken	436b792ca8a50415258542346dcdee4509c18cc5f507ecfef2eee1cb7b7ef7f4
lend	1dc383e25136d640724c4623d9c1c78b66dbe5253ebb50a81d982e4b06e2540c
log	3c8126345418df2225f1deb70108876c825b50cedecf59c5acac760e6d6aba90

2021-05-26:

Contract Name	Code Hash(eosio.cdt v1.6.3)

lend	87511c9dc0f96cea0d2b0d2f529744b9929b138810b5fbafe24e7f1d5bec53f3

4.2 主要合约函数权限分析

在审计过程中，慢雾安全团队对核心合约的可见函数进行权限分析，结果如下:

btoken		
Function Name	Visibility	Authority
create	Public	LEND_CONTRACT
remove	Public	-
issue	Public	st.issuer
retire	Public	st.issuer
transfer	Public	LEND_CONTRACT/from
reset	Public	ADMIN_ACCOUNT

lend		
Function Name	Visibility	Authority
borrow	Public	owner
stake	Public	owner
unstake	Public	owner
createtoken	Public	ADMIN_ACCOUNT
removetoken	Public	ADMIN_ACCOUNT
modifyconfig	Public	ADMIN_ACCOUNT
enableborrow	Public	ADMIN_ACCOUNT
enableascoll	Public	ADMIN_ACCOUNT
fix	Public	ADMIN_ACCOUNT
ontransfer::do_redeem	-	-
ontransfer::do_liquidation	-	_self
ontransfer::do_deposit	-	owner
ontransfer::do_repay	-	-

log		
Function Name	Visibility	Authority
depositlog	Public	LEND_CONTRACT
redeemlog	Public	LEND_CONTRACT
borrowlog	Public	LEND_CONTRACT

repaylog	Public	LEND_CONTRACT
liquidlog	Public	LEND_CONTRACT
stakelog	Public	LEND_CONTRACT
unstakelog	Public	LEND_CONTRACT

4.3 代码审计详情

4.3.1 中危漏洞

4.3.1.1 数值溢出风险

合约多个地方未使用 safemath 进行算术运算，有数值溢出风险。例如：

代码位置: lend.cpp

```
uint64_t decrease_usdt = price_usdt * quantity.amount / PRICE_BASE_PRECISION;
```

修复状态: 未修复

4.3.2 低危漏洞

4.3.2.2 权限未校验

lend.cpp 中 do_redeem/do_repay 函数未进行权限校验

修复状态: 已修复

4.3.3 增强建议

4.3.3.1 do_deposit 未实现还款

当充值的代币为借出的代币时，未实现自动还款。

代码位置: lend.cpp

```
void lend::do_deposit(const name& owner, const name& contract, const asset& quantity) {
require_auth(owner);
print_f("Deposit symbol: %\n", quantity.symbol);
check(quantity.amount > 0, "must deposit positive quantity");

auto reserve_id = get_reserve_id(contract, quantity.symbol);
auto r_itr = _reserves.find(reserve_id);
check(r_itr != _reserves.end(), "invalid token");
check(r_itr->is_active, "requires an inactive reserve");
check(!r_itr->is_freezed, "requires an unfreezed reserve");
check(r_itr->minimum_deposit <= quantity, "deposit amount limit");

auto exchange_rate = calc_btoken_exchange_rate(*r_itr);
auto btoken_quantity = asset(quantity.amount * 1e8 / exchange_rate, r_itr->bsym);
check(btoken_quantity.is_valid(), "Invalid btoken");
check(btoken_quantity.amount > 0, "Mint btoken amount error.");
print_f("btoken_quantity: %\n", btoken_quantity);

// update
auto btoken_balance = utils::get_balance(BTOKEN_CONTRACT, owner, r_itr->bsym);
update_state_on_deposit(r_itr, owner, quantity, btoken_balance.amount == 0);
btoken_balance += btoken_quantity;

auto reserve_balance = asset((uint128_t)(btoken_balance.amount) * exchange_rate / 1e8, r_itr->sym);

// issue btoken
action(
permission_level{_self, "active"_n},
BTOKEN_CONTRACT,
"issue"_n,
make_tuple(owner, btoken_quantity, string("mint btoken on deposit"))
).send();
// log
action(
permission_level{_self, "active"_n},
```

```
LOG_CONTRACT,
"depositlog"_n,
make_tuple(owner, reserve_id, r_itr->contract, quantity, btoken_quantity, btoken_balance,
reserve_balance, current_time_point())
).send();
}
```

修复状态: 充值和借出是独立的过程，产品支持低押借贷为同一种币。

5. 审计结果

5.1 总结

审计结论：主网合约未多签，有风险

审计编号：0X002103120001

审计时间：2021 年 03 月 12 日

复审时间：2021 年 06 月 02 日

审计团队：慢雾安全团队

审计总结：慢雾安全团队采用人工结合内部工具对代码进行分析。审计期间发现了 3 个问题。其中包含 0 个高危漏洞、1 个中危漏洞、1 个低危漏洞，并提出了 1 点增强建议。经过与项目方沟通反馈确认审计过程中发现的风险均已修复或在可承受范围内。

6. 声明

慢雾仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，慢雾无法判断其智能合约安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料(简称“已提供资料”)。慢雾假设：已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任。

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号