Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.04.05, the SlowMist security team received the OptionPanda team's security audit application for OptionPanda, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.

Level	Description
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability
- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit

- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Aduit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Option Panda allows option underwriting and trading in a decentralized setting.

Option buyers simply buy a call/put option on the platform, waiting for its settlement for prospective profit; option sellers simply deposit underlying asset to a pool to participate in a pooled option underwriting, collecting premiums paid by option buyers.

Initial audit files:

https://github.com/GiantPandaZoo/OptionPandaContract

commit: da2c4dbd714118056d8fcdc3b8e79fec9c05641f

Final audit files:

https://github.com/GiantPandaZoo/OptionPandaContract

commit: e16f46fab4448cec84655850bcc40777d4f4be31

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Missing event log	Others	Suggestion	Fixed

NO	Title	Category	Level	Status
N2	Missing permission control	Authority Control Vulnerability	Suggestion	Ignored
N3	Risk of excessive authority	Authority Control Vulnerability	Medium	Ignored
N4	Option counterfeiting issue	Others	Low	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

AggregateUpdater			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getNextUpdateTime	External	-	-
update	External	Can Modify State	-
addPools	External	Can Modify State	onlyOwner
removePool	Public	Can Modify State	onlyOwner

AggregateUpdater			
clearPools	Public	Can Modify State	onlyOwner

PandaBase			
Function Name	Visibility	Mutability	Modifiers
_slotSupply	Internal	-	-
_calcProfits	Internal	-	-
_sendProfits	Internal	Can Modify State	-
_totalPledged	Internal	-	-
<Constructor>	Public	Can Modify State	-
init	Public	Can Modify State	onlyOwner
owner	External	-	-
transferOwnership	External	Can Modify State	onlyOwner
pausePooler	External	Can Modify State	onlyOwner
unpausePooler	External	Can Modify State	onlyOwner
pauseBuyer	External	Can Modify State	onlyOwner
unpauseBuyer	External	Can Modify State	onlyOwner
optionsLeft	External	-	-
buy	External	Can Modify State	whenBuyerNotPaused
setOPAReward	External	Can Modify State	onlyOwner
_sigmaToIndex	Private	-	-

PandaBase			
premiumCost	Public	-	-
listOptions	External	-	-
currentUtilizationRate	External	-	-
getNextUpdateTime	Public	-	-
setRefreshPeriod	External	Can Modify State	-
update	Public	Can Modify State	-
_settleOption	Internal	Can Modify State	-
updateOPAReward	Internal	Can Modify State	-
updateSigma	Internal	Can Modify State	-
adjustSigma	External	Can Modify State	onlyOwner
checkPremium	External	-	-
claimPremium	External	Can Modify State	whenPoolerNotPaused
checkOPA	External	-	-
claimOPA	External	Can Modify State	whenPoolerNotPaused
settlePooler	External	Can Modify State	onlyPoolerTokenContract
_settlePooler	Internal	Can Modify State	-
NWA	Public	-	-
checkProfits	External	-	-
checkOptionProfits	Internal	-	-
claimProfits	External	Can Modify State	whenBuyerNotPaused

PandaBase			
settleBuyer	External	Can Modify State	onlyOptions
_settleBuyer	Internal	Can Modify State	-
setPoolManager	External	Can Modify State	onlyOwner
setUpdater	External	Can Modify State	onlyOwner
setOPAToken	External	Can Modify State	onlyOwner
setVestingContract	External	Can Modify State	onlyOwner
setUtilizationRate	External	Can Modify State	onlyOwner
setMaxUtilizationRate	External	Can Modify State	onlyOwner
getAssetPrice	Public	-	-

NativeCallOptionPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	PandaBase
name	Public	-	-
deposit	External	Payable	whenPoolerNotPaused
withdraw	External	Payable	whenPoolerNotPaused
_totalPledged	Internal	-	-
_calcProfits	Internal	-	-
_sendProfits	Internal	Can Modify State	-
_slotSupply	Internal	-	-

ERC20CallOptionPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	PandaBase
name	Public	-	-
depositAsset	External	Payable	whenPoolerNotPaused
withdrawAsset	External	Payable	whenPoolerNotPaused
_totalPledged	Internal	-	-
_sendProfits	Internal	Can Modify State	-
_calcProfits	Internal	-	-
_slotSupply	Internal	-	-

PutOptionPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	PandaBase
name	Public	-	-
depositUSDT	External	Payable	whenPoolerNotPaused
withdrawUSDT	External	Payable	whenPoolerNotPaused
_totalPledged	Internal	-	-
_sendProfits	Internal	Can Modify State	-
_calcProfits	Internal	-	-
_slotSupply	Internal	-	-

PandaFactory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createOption	External	Can Modify State	-
createPoolerToken	External	Can Modify State	-
getCDF	External	-	-
getUSDTContract	External	-	-

PoolerToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
mint	External	Can Modify State	onlyPool
burn	External	Can Modify State	onlyPool
getPool	Public	-	-
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-
transfer	Public	Can Modify State	-
allowance	Public	-	-

PoolerToken			
approve	Public	Can Modify State	-
transferFrom	Public	Can Modify State	-
increaseAllowance	Public	Can Modify State	-
decreaseAllowance	Public	Can Modify State	-
_transfer	Internal	Can Modify State	-
_mint	Internal	Can Modify State	-
_burn	Internal	Can Modify State	-
_approve	Internal	Can Modify State	-
_setupDecimals	Internal	Can Modify State	-
_beforeTokenTransfer	Internal	Can Modify State	-

Staking			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
stake	External	Can Modify State	-
claimRewards	External	Can Modify State	-
withdraw	External	Can Modify State	-
numStaked	External	-	-
totalStaked	External	-	-
checkReward	External	-	-

Staking			
setOPAReward	External	Can Modify State	onlyOwner
settleStaker	Internal	Can Modify State	-
updateOPAReward	Internal	Can Modify State	-

PandaView			
Function Name	Visibility	Mutability	Modifiers
getBuyerRounds	External	-	-
getPoolerRounds	External	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event log

Category: Others

Content

In the AggregateUpdater contract, the owner can add or remove pools through the addPools function and removePool function, but no event recording is performed.

In the PandaFactory contract, users can create Option and PoolerToken through the createOption function and the createPoolerToken function, but the event is not recorded.

In the staking contract, the owner can modify the block reward number through the setOPAReward function, but the event is not recorded.

Solution

It is recommended to record events for sensitive operations for follow-up review.

Status

Fixed

[N2] [Suggestion] Missing permission control

Category: Authority Control Vulnerability

Content

In the PandaFactory contract, any user can create Option and PoolerToken through the createOption function and the createPoolerToken function.

```
    function createOption(uint duration_, uint8 decimals_, IOptionPool poolContract)
external override
        returns (IOption option) {
        return new Option(duration_, decimals_, poolContract);
    }

    function createPoolerToken(uint8 decimals_, IOptionPool poolContract) external
override returns (IPoolerToken poolerToken) {
        return new PoolerToken(decimals_, poolContract);
    }
```

Solution

It is recommended to control permissions, which can only be called by the Pool contract.

Status

Ignored; The project party has checked the Option contract in the Pool contract. This arbitrary creation problem does not affect the overall business security.

[N3] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the OptionPandaToken contract, the owner can mint any tokens without restriction through the mint function. The owner can release any user's locked tokens through the release function. This will lead to the risk of excessive owner authority.

```
function mint(address account, uint256 amount) public onlyOwner {
    _mint(account, amount);

}
```

```
function release(address account, uint256 releaseAmount) public onlyOwner {
    require(account != address(0), "OPA: release zero address");

    TimeLock storage timelock = _timelock[account];
    timelock.amount = timelock.amount.sub(releaseAmount);
    if(timelock.amount == 0) {
        timelock.releaseTime = 0;
    }
}
```

Solution

It is recommended to mint enough tokens and remove this function when the contract is initialized, or set an upper limit for token minting, or transfer the minting authority to community governance. And check whether the current time is greater than the unlocking time during release.

Status

Ignored

[N4] [Low] Option counterfeiting issue

Category: Others

Content

In the Pool contract, the user can buy options through the buy function, but the buy function does not check whether the incoming option contract is in the Pool's Options list, which will lead to option rights counterfeiting problems.

```
    function buy(uint amount, IOption optionContract, uint round) external override
whenBuyerNotPaused {
        // check if option current round is the given round, we cannot buy previous
rounds
        require (optionContract.getRound() == round, "expired");
        // cap amount to remaing options
        if (optionContract.balanceOf(address(this)) < amount) {
            amount = optionContract.balanceOf(address(this));
        }

        // calculate premium cost
        uint premium = premiumCost(amount, optionContract);

        // transfer premium USDTs to this pool
        USDTContract.safeTransferFrom(msg.sender, address(this), premium);

        // transfer options to msg.sender
        optionContract.safeTransfer(msg.sender, amount);

        // credit premium to option contract
        optionContract.addPremium(msg.sender, premium);

        // log
        emit Buy(msg.sender, address(optionContract), round, amount, premium);

        // entropy storage to refund to update() caller
        entropy.push(block.number);
```

```
        // OPTION PARAMETERS MAINTENANCE
        // sigma: count sold options
        _sigmaSoldOptions = _sigmaSoldOptions.add(amount);

        // other periodical refresh
        if (block.timestamp > _nextRefresh) {
            updateSigma();
        }

        // open new round, one caller will get refund for an effective update
        if (block.timestamp > getNextUpdateTime()) {
            update();
        }
    }
```

Solution

It is recommended to check whether the purchased option is in the Pool's options list during the buy operation.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002104140003	SlowMist Security Team	2021.04.05 - 2021.04.13	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk, 2 suggestion vulnerabilities. And 1 medium risk, 1 suggestion vulnerabilities were ignored, all other findings were fixed. The code was not deployed to the mainnet.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist