Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2022.09.22, the SlowMist security team received the X2Y2 team's security audit application for X2Y2 - NFT Lending, 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.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team 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.

Level	Description
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:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Audit version:

contracts.zip: 43fdfed10c92179052eb7a24238ad7cfcd7bded1d2a62bc93bcdb45414f01d4b

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Risk of excessive authority	Authority Control Vulnerability	High	Fixed
N2	Token compatibility issue	Others	Suggestion	Ignored
N3	Dev address setting enhancement suggestions	Others	Suggestion	Ignored

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

X2Y2 - NFT Lending	
Contract Name	Contract Address
XY3	0xc28f7ee92cd6619e8eec6a70923079fbafb86196
Delegate	0xef887e8b1c06209f59e8ae55d0e625c937344376

X2Y2 - NFT Lending	
Xy3Nft	0x0e258c84df0f8728ae4a6426ea5fd163eb6b9d1b

4.2 Visibility Description

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

Config			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
pause	External	Can Modify State	onlyRole
unpause	External	Can Modify State	onlyRole
updateMaxBorrowDuration	External	Can Modify State	onlyRole
updateMinBorrowDuration	External	Can Modify State	onlyRole
updateAdminShare	External	Can Modify State	onlyRole
updateAdminFeeReceiver	External	Can Modify State	onlyRole
setERC20Permits	External	Can Modify State	onlyRole
setERC721Permits	External	Can Modify State	onlyRole
getERC20Permit	Public	-	-
getERC721Permit	Public	-	-
supportsInterface	Public	-	-
_setERC20Permit	Internal	Can Modify State	-
_setERC721Permit	Internal	Can Modify State	-

Delegate			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
erc20Transfer	External	Can Modify State	onlyRole
erc721Transfer	External	Can Modify State	onlyRole

LoanStatus			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
createLoan	Internal	Can Modify State	-
resolveLoan	Internal	Can Modify State	-
getLoanState	Public	-	-

Xy3Nft			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721
burn	External	Can Modify State	onlyRole
mint	External	Can Modify State	onlyRole
setBaseURI	External	Can Modify State	onlyRole
supportsInterface	Public	-	-
exists	External	-	-

Xy3Nft			
_getChainID	Internal	-	-
_baseURI	Internal	-	-
_setBaseURI	Internal	Can Modify State	-

XY3			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Config LoanStatus
borrow	External	Can Modify State	whenNotPaused nonReentrant
repay	External	Can Modify State	nonReentrant loanIsOpen
liquidate	External	Can Modify State	nonReentrant loanIsOpen
cancelByNonce	External	Can Modify State	-
cancelByTimestamp	External	Can Modify State	-
getRepayAmount	External	-	-
getNonceUsed	External	-	-
getTimestampCancelled	External	-	-
_resolveLoan	Internal	Can Modify State	-
_loanSanityChecks	Internal	-	-
_getPartiesAndData	Internal	-	-
_payoffAndFee	Internal	-	-
_borrow	Internal	Can Modify State	-

XY3			
_checkSignatures	Private	-	-
_createLoanDetail	Internal	-	-
_loanMaturityDate	Private	-	-

4.3 Vulnerability Summary

[N1] [High] Risk of excessive authority

Category: Authority Control Vulnerability

Content

1.In the Config contract, the owner role can add manager role and can update the adminShare, and the manager role can set or remove the ERC20 and ERC721 tokens through the setERC20Permits and setERC721Permits functions. If the ERC20 and ERC721 on the loan list, the manager sets these permit as false may cause the risk of excessive authority.

Code location:

Config.sol#129-142, 165-198, 249-268

```
function updateAdminShare(uint16 _newAdminShare)
        external
        override
        onlyRole(MANAGER_ROLE)
    {
        require(
            _newAdminShare <= HUNDRED_PERCENT,
            "basis points > 10000"
        );
        if(adminShare != _newAdminShare) {
            adminShare = _newAdminShare;
            emit AdminFeeUpdated(_newAdminShare);
        }
    }
```

```
function setERC20Permits(address[] memory _erc20s, bool[] memory _permits)
    external
    override
    onlyRole(MANAGER_ROLE)
{
    require(
        _erc20s.length == _permits.length,
        "address and permits length mismatch"
    );

    for (uint256 i = 0; i < _erc20s.length; i++) {
        _setERC20Permit(_erc20s[i], _permits[i]);
    }
}

function setERC721Permits(address[] memory _erc721s, bool[] memory _permits)
    external
    override
    onlyRole(MANAGER_ROLE)
{
    require(
        _erc721s.length == _permits.length,
        "address and permits length mismatch"
    );

    for (uint256 i = 0; i < _erc721s.length; i++) {
        _setERC721Permit(_erc721s[i], _permits[i]);
    }
}

function _setERC20Permit(address _erc20, bool _permit) internal {
    require(_erc20 != address(0), "erc20 is zero address");

    erc20Permits[_erc20] = _permit;

    emit ERC20Permit(_erc20, _permit);
}

function _setERC721Permit(address _erc721, bool _permit) internal {
    require(_erc721 != address(0), "erc721 is zero address");

    erc721Permits[_erc721] = _permit;
```

```
        emit ERC721Permit(_erc721, _permit);
    }
```

2.In the Xy3Nft contract, the owner role can add minter and signer role, and the minter role can mint and burn the Xy3Nft tokens through the burn and mint fucntions. Every tickets mint by the XY3 are bound with a loanId and the loan state. If the minter burn the Xy3Nft with a loanId, the repay and liquidate will fail, the borrower's NFT will locked in the contract and the lender will not get the repayment.

Code location:

Xy3Nft.sol#63-85

```
    function burn(uint256 _tokenId) external onlyRole(MINTER_ROLE) {
        delete tickets[_tokenId];
        _burn(_tokenId);
    }

    function mint(
        address _to,
        uint256 _tokenId,
        bytes calldata _data
    ) external onlyRole(MINTER_ROLE) {
        require(_data.length > 0, "no data");

        uint256 loanId = abi.decode(_data, (uint256));
        tickets[_tokenId] = Ticket({loanId: loanId, minter: msg.sender});
        _safeMint(_to, _tokenId, _data);
    }
```

Solution

It is recommended to use a time lock mechanism or community governance to restrict.

Status

Fixed; After communication with the project team, they expressed that the minter role of the Xy3Nft contract is the XY3 contract, and the admin role of the Xy3Nft contract has been transferred to the timelock contract:

0x3c113749BAC6FFccC7A36B63D6c64Ce645D50d6d, which is controlled by a gnosis multisig wallet:

0x0b67C19b1CE45D363cCABa8307a3d83a8fa6AB77.

[N2] [Suggestion] Token compatibility issue

Category: Others

Content

In the XY3 contract, the lender will transfer the ERC20 token to the borrower in the borrow function also to the lender and adminFeeReceiver in the repay function. And this transfer is used the SafeER20 safeTransferFrom function and transfer the exact amount of the borrowAmount, payoffAmount and adminFee. If the borrowAsset ERC20 tokens are the deflationary tokens (or other tokens that require a transfer fee) which will cause the call failed.

Code location:

XY3.sol#114,117,314

Delegate.sol#20-22

```
        IDelegate(delegate).erc20Transfer(msg.sender, lender, loan.borrowAsset,
payoffAmount);

        // Transfer admin fee
        IDelegate(delegate).erc20Transfer(msg.sender,
adminFeeReceiver,loan.borrowAsset, adminFee);

        IDelegate(delegate).erc20Transfer(_lender, msg.sender,
_loanDetail.borrowAsset, _loanDetail.borrowAmount);

    function erc20Transfer(address sender, address receiver, address token, uint256
amount) external override  onlyRole(DELEGATION_CALLER){
        IERC20(token).safeTransferFrom(sender, receiver, amount);
    }
```

Solution

It is recommended to record the difference between the contract balance before and after the token transfer as the actual transfer amount. Or do not permit such ERC20 tokens in the token list.

Status

Ignored; After communication with the project team, they expressed that only the standard ERC20 tokens permitted will be allowed. And for now, only weth is used.

[N3] [Suggestion] Dev address setting enhancement suggestions

Category: Others

Content

In the Config contract, the manager role can update the adminFeeReceiver in the updateAdminFeeReceiver function to receive the adminFee. If the adminFeeReceiver address is an EOA address, in a scenario where the private key is leaked, the team's revenue will be stolen.

Code location:

Config.sol#148-158

```
    function updateAdminFeeReceiver(address _newAdminFeeReceiver)
        external
        override
        onlyRole(MANAGER_ROLE)
    {
        require(_newAdminFeeReceiver != address(0), "Invalid receiver address");
        if(adminFeeReceiver != _newAdminFeeReceiver) {
            adminFeeReceiver = _newAdminFeeReceiver;
            emit AdminFeeReceiverUpdated(adminFeeReceiver);
        }
    }
```

Solution

It is recommended to set the owner address as a multi-signature contract to avoid the leakage of private keys and the theft of team rewards.

Status

Ignored; After communication with the project team, they expressed that the MANAGER_ROLE can only be granted by admin, and admin role will be transferred to a timelock contract later.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002209290002	SlowMist Security Team	2022.09.22 - 2022.09.29	Passed

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 high risk, 2 suggestion vulnerabilities. And 2 suggestion vulnerabilities were ignored; All other findings were fixed.

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