


SECURITY AUDIT REPORT for
PINE PROTOCOL

Jan 28, 2022 - Nassec Verified 

Executive Summary

This smart contract audit report is prepared for [ERC721LendingPoolETH01.sol](#), [VerifySignaturePool01.sol](#), & [Pinewallet01.sol](#), smart contracts of [Pine Protocol](#), after a successful functionality testing, source code review and black box application security penetration exercise.

Type

Smart Contract

Languages

Solidity

Timeline

Jan 09 , 2022 to Jan 28, 2022

Methodology

White Box Testing & Black Box Testing

Change log

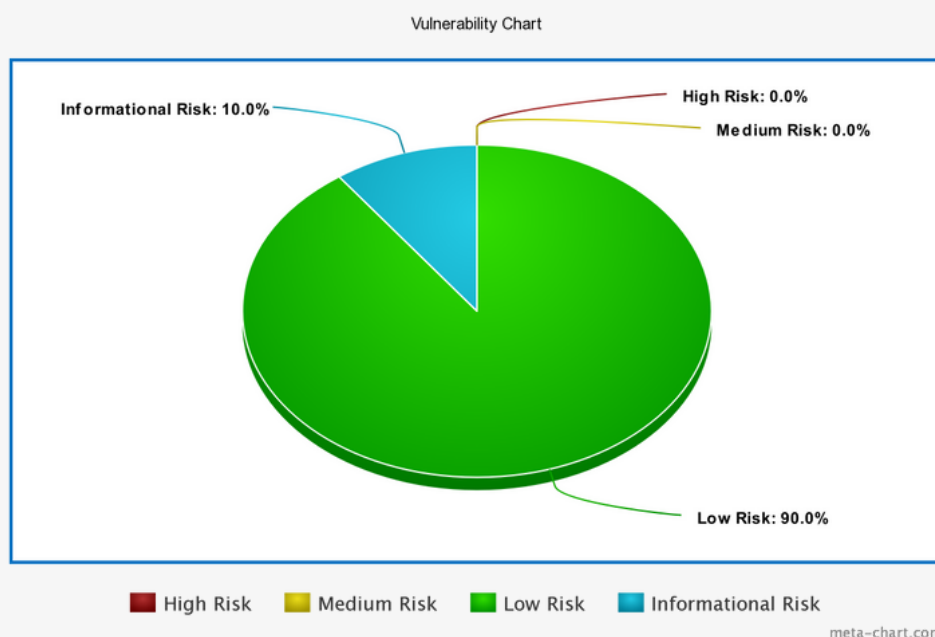
Jan 07, 2022 - Requirement received via github - commit history -

2b708dae0cf03990b9b31f7893bd3eba8cfb2ad8

Jan 09, 2022 - Beginning of first phase of audit.

Jan 26, 2022 - Completion of first phase of audit, report submission.

Jan 28, 2022 - Retest; Final Report Submission.



High Risk

Vulnerabilities that can be exploited publicly, workaround or fix/ patch available by the vendor.

Medium Risk

Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Patch/ workaround not yet released.

Undetermined Risk

Vulnerabilities that has uncertain impact.

0

0

0

Low Risk

Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Vulnerability observed may not have a high rate of occurrence. Patch/ workaround released by the vendor.

Informational Risk

Vulnerabilities that have the minimum impact on the system.

9

1

Summary of Findings

| Vulnerability | Risk | Status |
|--|---------------|--------|
| 1. Initialize a constructor instead of a function at line 37 | Low | Open |
| 2. Use double quotes for string literals at line 41 | Informational | Fixed |
| 3. Unlocked Pragma in PineWallet01.sol at line 3 | Low | Fixed |
| 4. Unlocked Pragma in VerifySignaturePool01.sol at line 2 | Low | Fixed |
| 5. Initialize a constructor instead of a function at line 47 | Low | Open |
| 6. Checks-Effects-Interactions pattern not implement in depositCollateral() function at line 44. | Low | Fixed |
| 7. Checks-Effects-Interactions pattern not implement in removeCollateral() function at line 49. | Low | Fixed |
| 8. Checks-Effects-Interactions pattern not implement in borrowETH() function at line 148. | Low | Fixed |
| 9. Checks-Effects-Interactions pattern not implement in liquidateLoan() function at line 268. | Low | Fixed |
| 10. Checks-Effects-Interactions pattern not implement in repayETH() function at line 213. | Low | Fixed |

Objective

The objective of the security audit of smart contract of *Pine Protocol* was to assess the state of security and uncover vulnerabilities in the smart contract and provide with a detail report comprising remediation strategy and recommendations to help mitigate the identified vulnerabilities and risks during the activity.

The assessment was conducted using two methods; source code review and black box testing.

Scope & Requirements

The technical scope for the conducted application security testing activity was restricted to:

- ERC721LendingPoolETH01.sol
- VerifySignaturePool01.sol
- Pinewallet01.sol

Files url

<https://github.com/pinedefi/lending-borrowing-smart-contracts>
Git commit hash - 2b708dae0cf03990b9b31f7893bd3eba8cfb2ad8

Provided Date

Jan 07, 2022; Email Access given via Github.

Methodology

White Box Testing/Source Code Review

Understand Program Specification

Understanding programming language used and coding practices followed is a key in performing efficient source code review solution. Our security researchers begin source code review by first understanding the program specification.

Obtain and Review Source Code

Our security researchers then co-ordinate with the development team to obtain the source code and start reviewing the codes line by line to identify flaws.

Identify Flaws

We apply a rigorous approach to identify flaws. That is we adhere to strict standard practices and retest major functions again to make sure we don't miss a single loophole.

Reporting

A documentation of where the flaws has been found and how the patches can be done will be submitted to the development team for a fix.

Black Box Testing

Information Gathering

Understanding how your system works is a key in providing you the right security solution. We do proper recon on the target and get as much information as we can. We extract details like OS, versions , framaeworks, Programming languages, open ports and overall details which will help us in vulnerability analysis.

Vulnerability Analysis

A holistic analysis of vulnerabilities present in your system. We first go through OWSAP top 10 vulnerability and after that we will look for possible vulnerability including logical issues which can expose to great security vulnerability.

Exploitation

With a hacker's mind our security researchers exploit your system to detect the last remaining vulnerability.

Reporting

A documentation of where and how the patches can be done will be submitted to your developers team for a fix. The document explains the root cause, and includes a remediation plan categorized on the basis of severity of the vulnerabilities.

Findings from the assessment

We found the following findings after performing the security audit of the smart contracts of *Pine Protocol* through source code review and black box testing.

Findings from Manual Testing

1. Initialize a constructor instead of a function at line 37

Description

In PineWallet01.sol, we have a function `initialize()` which when called assigns the owner of the smart contract and mints 1 token. This function can be called by anyone once.

Impact

Unknown

Risk

Low

Status

Open

Proof of Concept

```
function initialize() public initializer {
    _safeMint(msg.sender, 1);
    __Ownable_init();
    __ERC721_init("PINEWALLET", "PINEWALLET");
    _baseURIextended = 'https://pinewallet-api.pine.loans/meta';
}
```

Remediation

Making a constructor which will be directly executed while deploying smart contract.

```
constructor() initializer {
    _safeMint(msg.sender, 1);
    __Ownable_init();
    __ERC721_init("PINEWALLET", "PINEWALLET");
    _baseURIextended = "https://pinewallet-api.pine.loans/meta";
}
```

2. Use double quotes for string literals at line 41

Description

In PineWallet01.sol, we have a function `initialize()` on which `_baseURIextended = 'https://pinewallet-api.pine.loans/meta'`

Impact

Decreases code quality

Risk

Informational

Status

Fixed

Proof of Concept

```
function initialize() public initializer {
    _safeMint(msg.sender, 1);
    __Ownable_init();
    __ERC721_init("PINEWALLET", "PINEWALLET");
    _baseURIextended = 'https://pinewallet-api.pine.loans/meta';
}
```

Remediation

Change single quote to double quotes

```
constructor() initializer {
    _safeMint(msg.sender, 1);
    __Ownable_init();
    __ERC721_init("PINEWALLET", "PINEWALLET");
    _baseURIextended = "https://pinewallet-api.pine.loans/meta";
}
```


3. Unlocked Pragma in PineWallet01.sol at line 3

Description

The PineWallet01.Sol file contains unlocked pragma. The use of unlocked pragma implies that the compiler will only use the specified version which brings inconsistency and unexpected behaviour in the future.

Impact

Brings inconsistency

Risk

Low

Status

Fixed

Proof of Concept

```
pragma solidity ^0.8.3;
```

Remediation

Remove Caret (^) from pragma and specify solidity version.

```
pragma solidity 0.8.3;
```

4. Unlocked Pragma in VerifySignaturePool01.sol at line 2

Description

The VerifySignaturePool01.Sol file contains unlocked pragma. The use of unlocked pragma implies that the compiler will only use the specified version which brings inconsistency and unexpected behaviour in the future.

Impact

Brings inconsistency

Risk

Low

Status

Fixed

Proof of Concept

```
pragma solidity ^0.8.3;
```

Remediation

Remove Caret (^) from pragma and specify solidity version.

```
pragma solidity 0.8.3;
```

5. Initialize a constructor instead of a function at line 47

Description

In ERC721LendingPoolETH01.sol, we have a function `initialize()` which when called assigns the owner of the smart contract. This function can be called by anyone once.

Impact

Unknown

Risk

Low

Status

Open

Proof of Concept

```
function initialize(address supportedCollection) public initializer {
    __Ownable_init();
    __Pausable_init();
    _supportedCollection = supportedCollection;
}
```

Remediation

Making a constructor which will be directly executed while deploying smart contract.

```
constructor(address supportedCollection) initializer {
    __Ownable_init();
    __Pausable_init();
    _supportedCollection = supportedCollection;
}
```

Findings from Automated Testing Using Slither

6. Checks-Effects-Interactions pattern not implement in depositCollateral() function at line 44.

Description

In PineWallet01.sol, depositCollateral function has not implemented Checks-Effects-Interactions pattern.

Impact

Reentrancy attack might be possible.

Risk

Low

Status

Fixed

Proof of Concept

External calls:

```
ERC721EnumerableUpgradeable(target).safeTransferFrom(msg.sender, address(this), nftID);
```

State variables written after the call:

```
collateralizedCollections[target] = true;
```

Remediation

Use Checks-Effects-Interactions pattern.

7. Checks-Effects-Interactions pattern not implement in removeCollateral() function at line 49.

Description

In PineWallet01.sol, removeCollateral function has not implemented Checks-Effects-Interactions pattern.

Impact

Reentrancy attack might be possible.

Risk

Low

Status

Fixed

Proof of Concept

External calls:

```
ERC721EnumerableUpgradeable(target).safeTransferFrom(address  
(this), msg.sender, nftID);
```

State variables written after the call:

```
collateralizedCollections[target] = false;
```

Remediation

Use Checks-Effects-Interactions pattern.

8. Checks-Effects-Interactions pattern not implemented in borrowETH() function at line 148.

Description

In ERC721LendingPoolETH01.sol, borrowETH() function has not implemented Checks-Effects-Interactions pattern.

Impact

Reentrancy attack might be possible.

Risk

Low

Status

Fixed

Proof of Concept

External calls:

```
IERC721(_supportedCollection).safeTransferFrom(
    msg.sender,
    address(this),
    nftID
);
```

Event emitted after the call:

```
emit LoanInitiated(
    msg.sender,
    _supportedCollection,
    nftID,
    _loans[nftID]
);
```

Remediation

Use Checks-Effects-Interactions pattern.

9. Checks-Effects-Interactions pattern not implement in liquidateLoan() function at line 268.

Description

In ERC721LendingPoolETH01.sol, liquidateLoan() function has not implemented Checks-Effects-Interactions pattern.

Impact

Reentrancy attack might be possible.

Risk

Low

Status

Fixed

Proof of Concept

External calls:

```
IERC721(_supportedCollection).safeTransferFrom(
    address(this),
    owner(),
    nftID
);
```

Event emitted after the call:

```
emit LoanTermsChanged(
    _loans[nftID].borrower,
    _supportedCollection,
    nftID,
    oldLoanTerms,
    _loans[nftID]
);
```

Remediation

Use Checks-Effects-Interactions pattern.

10. Checks-Effects-Interactions pattern not implemented in repayETH() function at line 213.

Description

In ERC721LendingPoolETH01.sol, repayETH() function has not implemented Checks-Effects-Interactions pattern.

Impact

Reentrancy attack might be possible.

Risk

Low

Status

Fixed

Proof of Concept

External calls:

```
IERC721(_supportedCollection).safeTransferFrom(
    address(this),
    _loans[nftID].borrower,
    nftID
);
```

```
msg.sender.call{value: toBeTransferred}("");
```

Event emitted after the call:

```
emit LoanTermsChanged(
    _loans[nftID].borrower,
    _supportedCollection,
    nftID,
    oldLoanTerms,
    _loans[nftID]
);
```

Remediation

Use Checks-Effects-Interactions pattern.

Conclusion

We found a total of 10 vulnerabilities in [ERC20_01.sol](#), [ERC721LendingPoolETH01.sol](#), [VerifySignaturePool01.sol](#), & [Pinewallet01.sol](#), out of which nine possessed low risk and one possessed informational risk. On the basis of our submission, the product development team of [Pine Protocol](#) has acknowledged and fixed 8 issues, while 2 issues have been left open.

In addition to this audit, we encourage the team to organize a public vulnerability disclosure program and adopt other necessary measures to minimize cyber risks associated with smart contracts.

Disclaimer

Considering the evolving nature of risks and vulnerabilities associated with solidity language, smart contracts and Ethereum network, Nassec cannot wholly guarantee that the contract will be free of vulnerabilities after the audit. We encourage the team to organize a public vulnerability disclosure program and adopt other necessary measures to minimize cyber risks associated with smart contracts.

About Nassec

Nassec is a cyber security firm dedicated to providing vulnerability management solutions. Nassec has been trusted by industries ranging from fintech to blockchain and SAAS to Ecommerce. With cyber attacks rising rapidly, businesses require cyber security solutions more than ever. Our tailor-made vulnerability management solutions help prevent cyber attacks. Our team consists of self-taught professional cyber security practitioners who follow international standard security protocols and possess strong work ethics. Our team has been recognized by tech giants such as Facebook, Microsoft, Sony, Etsy and others for contributing to make them more secure.

National Agency Security System Pvt. Ltd. (Nassec)

Address - Kathmandu, Nepal

Email - info@nassec.io